A Survey on Numerical Methods for Unconstrained Optimization Problems

by

CHUNG Shun Shing

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Mathematics

August 2002

Abstract

The abstract of the thesis entitled:

A Survey on Numerical Methods for Unconstrained Optimization Problems

submitted by CHUNG Shun Shing

for the degree of Master of Philosophy

at The Chinese University of Hong Kong in August 2002

Optimization methods, or methods for computing the minimum of a function of several variables, are very important in the 21st century. They can be applied to solve problems from many practical applications, and they can provide the numerical analysts with a tool tackling a wide range of non-linear problems. There is a vast literature devoted to the development of optimization methods in recent years.

This thesis has two main objectives: the first is to give a comprehensive and detailed survey on the current numerical methods available for unconstrained optimization; and the second is to provide the reader with a framework to help him follow future developments. Optimization is essentially a practical tool, principally used by non-mathematicians; in contrast, most research publications in optimization seems to be written in a style that is only intelligible to mathematicians. This thesis attempts to bridge the gap. Little mathematical knowledge is assumed on the part of the reader. The first two chapters outline the background and basic theory necessary to understand the subsequent chapters; and the remaining chapters introduce some existing optimization methods: steepest descent method, Newton's method, conjugate gradient method, Quasi-Newton methods, and the like, which the reader would be familiar with. Finally, the advantages and limitations of each method are discussed.

摘要

香港中文大學碩士論文摘要

論文題目：無約束最優化問題的數值方法

鍾信成　　二零零二年八月

最優化是一門應用相當廣泛的學科，它討論決策問題的最佳選擇之特性，構造尋求最佳解的計算方法，研究這些計算方法的理論性質及實際計算表現。隨著計算機的高速發展和優化計算方法的進步，規模越來越大的優化問題得以解決。因為最優化問題廣泛見於經濟計劃、工程設計、生產管理、交通運輸、國防等重要領域，它已受到各國政府部門、科研機構和產業部門的高度重視。

再者，最優化理論與方法也是一門應用很強的學科。它研究某些數學上定義的問題的最優解，即對於給出的實際問題，從眾多的方案中選出最優方案。

本篇論文全面地、系統地介紹了最優化理論與方法，詳細論述了無約束最優化的最優性條件、求解方法以及各類求解方法的特點，包括其優點和限制。

本篇論文首兩章介紹最優化理論與方法之背景和一些數學基礎；而在餘下的章節中，會深入介紹一些現存的無約束最優化求解方法，包括：最速下降法、牛頓法、共軛梯度法、擬牛頓法等方法，並會探討它們的收斂性，給出全局收斂性和局部收斂性的結果。

ACKNOWLEDGMENTS

I wish to express my sincere and deepest gratitude to my supervisor Prof. ZOU Jun for his inspired guidance, insight suggestions, constant encouragement, patience and help during the course of my Master of Philosophy studies. And I would also like to thank Prof. DAI Yu Hong for sharing with me the the useful papers and articles for the Optimization Problems and giving very helpful opinions to me.

Also, special thanks goes to all teaching staffs, supporting staffs and postgraduate students of the Department of Mathematics, the Chinese University of Hong Kong. They give big and kindly help to me during the past two years of my postgraduate studies.

Moreover, I would like to thank my colleagues (in Numerical Analysis Group) Mr. CHAN Kit Hung Michael, Mr. CHAN Yuet Tai, Mr. CHUNG Tsz Shun Eric, Mr. LAU Kin Wing, Miss CHAN Kai Yam, Mr. XIE Jian Li, Miss CHENG Ting, Mr. WONG Chak Fu Jeff, Mr. CHEN Yong, Mr. CHUNG Hau Leung Eric, Mr. TAM Yue Hung, Mr. WONG Chi Yan Michael and Mr. LING Kai Tung Russ for their very helpful discussions and suggestions through the group meetings and the casual talking.

Lastly, my family and intimates who have offered me special concern and encouragement to overcome all the ups and downs, are undoubtedly worthy of my deepest thanks.

CHUNG Shun Shing Samson

The Chinese University of Hong Kong

August 2002

Contents

List of Figures x

1 Introduction 1

1.1 Background and Historical Development 1

1.2 Practical Problems 3

1.2.1 Statistics 3

1.2.2 Aerodynamics 4

1.2.3 Factory Allocation Problem 5

1.2.4 Parameter Problem 5

1.2.5 Chemical Engineering 5

1.2.6 Operational Research 6

1.2.7 Economics 6

1.3 Mathematical Models for Optimization Problems . . 6

1.4 Unconstrained Optimization Techniques 8

1.4.1 Direct Method - Differential Calculus 8

1.4.2 Iterative Methods 10

1.5 Main Objectives of the Thesis 11

2 Basic Concepts in Optimizations of Smooth Functions 14

2.1 Notation 14

2.2 Different Types of Minimizer 16

2.3 Necessary and Sufficient Conditions for Optimality . 18

2.4 Quadratic Functions 22

2.5 Convex Functions 24

2.6 Existence, Uniqueness and Stability of a Minimum . . 29

2.6.1 Existence of a Minimum 29

2.6.2 Uniqueness of a Minimum 30

2.6.3 Stability of a Minimum 31

2.7 Types of Convergence 34

2.8 Minimization of Functionals 35

3 Steepest Descent Method **37**

3.1 Background 37

3.2 Line Search Method and the Armijo Rule 39

3.3 Steplength Control with Polynomial Models 43

3.3.1 Quadratic Polynomial Model 43

3.3.2 Safeguarding 45

3.3.3 Cubic Polynomial Model 46

3.3.4 General Line Search Strategy 49

3.3.5 Algorithm of Steepest Descent Method 51

3.4 Advantages of the Armijo Rule 54

3.5 Convergence Analysis 56

4 Iterative Methods Using Second Derivatives **63**

4.1 Background 63

4.2 Newton's Method 64

4.2.1 Basic Concepts 64

4.2.2 Convergence Analysis of Newton's Method . . 65

4.2.3 Newton's Method with Steplength 69

4.2.4 Convergence Analysis of Newton's Method with Step-length 70

4.3 Greenstadt's Method 72

4.4 Marquardt-Levenberg Method 74

4.5 Fiacco and McComick Method 76

4.6 Matthews and Davies Method 79

4.7 Numerically Stable Modified Newton's Method 80

4.8 The Role of the Second Derivative Methods 89

5 Multi-step Methods **92**

5.1 Background 93

5.2 Heavy Ball Method 94

5.3 Conjugate Gradient Method 99

5.3.1 Some Types of Conjugate Gradient Method . 99

5.3.2 Convergence Analysis of Conjugate Gradient Method 108

5.4 Methods of Variable Metric and Methods of Conjugate Directions 111

5.5 Other Approaches for Constructing the First-order Methods 116

6 Quasi-Newton Methods **121**

6.1 Disadvantages of Newton's Method 122

6.2 General Idea of Quasi-Newton Method 124

6.2.1 Quasi-Newton Methods 124

6.2.2 Convergence of Quasi-Newton Methods 129

6.3 Properties of Quasi-Newton Methods 131

6.4 Some Particular Algorithms for Quasi-Newton Methods 137

6.4.1 Single-Rank Algorithms 137

6.4.2 Double-Rank Algorithms 144

6.4.3 Other Applications 149

6.5 Conclusion 152

7 Choice of Methods in Optimization Problems **154**

7.1 Choice of Methods 154

7.2 Conclusion 157

Bibliography 158

List of Figures

2.1 A function $f(x)$ of single variable. 18

2.2 An example with a discontinuous function $f(x)$. 19

2.3 A strictly convex function. 25

2.4 A convex function. 26

3.1 The sequence $f(x_n) = x_n^2$ is strictly decreasing, but does not converge to the minimizer $(0, 0)$. 41

3.2 Permissible values of α under Armijo rule 42

3.3 Backtracking at the first iteration, using the quadratic model . . . 45

3.4 $q(\alpha)$ has a unique minimizer when $\xi'' > 0$ 48

3.5 $q(\alpha)$ has a unique minimizer when $\xi'' < 0$ 49

4.1 Divergence of Newton's method. 68

5.1 The comparison between the Heavy Ball Method and Steepest Descent Method 95

Chapter 1

Introduction

"The optimist proclaims that we live in the best of all possible worlds; and the pessimist fears this true."

J. B. Cabell

1.1 Background and Historical Development

The fundamental problem of optimization is to arrive at the best possible decision in any given set of circumstances. Of course, many situations arise where the 'best' is unattainable for one reason or another; sometimes what is 'best' for one person is 'worst' for another; more often we are not at all sure what is meant by 'best'. Therefore, the first step in a mathematical optimization problem is to choose some quantity, typically a function of several variables, to be maximized or minimized, subject possibly to one or more constraints. The commonest types of constraints are equalities and inequalities which must be satisfied by the variables of the problem, but many other types of constraint are possible, for example, a solution in integers may be required.

The next step is to choose a mathematical method to solve the optimization problem. Such methods are usually called optimization techniques, or algorithms.

According to [120], [76] and [47], the choice of algorithm is by no means obvious, for the theory and practice of optimization has developed rapidly since the advent of electronic computers in 1945. It came of age as a subject in the mathematical curriculum in the 1950's, when the well-established methods of the differential calculus and the calculus of variations were combined with the highly successful new techniques of mathematical programming which were being developed at that time. The programmers, it was said, had joined forces with the hillclimbers.

The optimization problems that have been posed and solved in recent years have tended to become more and more elaborate, not to say abstract. Perhaps the most outstanding example of the rapid development of optimization techniques occurred with the introduction of dynamic programming by Bellman in 1957 and of the maximum principle by Pontryagin in 1958. These techniques were designed to solve the problem of the optimal control of dynamical systems. Both dynamic programming and the maximum principle are closely related to the calculus of variations, and hence to each other.

The simply-stated problem of maximizing or minimizing a given function of several variables has attracted the attention of many mathematicians over the past twenty-five years or so. The direct search methods of solution, which involve function evaluations and comparisons only, are usually simpler, though less accurate for the same computational effort, than the indirect or gradient methods, which require values of the function and its derivatives. Both types of method

are still undergoing development, with the major emphasis being on the search for efficient and reliable algorithms to deal with general non-linear functions.

For the sake of simplicity, and following the historical development of the subject, most of the theory of the gradient methods in this thesis is restricted to the case of quadratic functions. The theory for non-quadratic functions is considerably more difficult and is the subject of the current research. However, the algorithms based on quadratic theory are usually successful when applied to non-quadratic functions.

1.2 Practical Problems

Many different types of practical problem will be considered as illustrations of the different optimization techniques. Some of these will be described in the following to indicate the types of problem that have been solved by these methods. Most of the problems in this section can be found in [120] and [37].

1.2.1 Statistics [120]

The frequency function of a population is completely determined once its parameters are known. For example, the binomial distribution is completely determined the parameters n, which is the number of independent trials of an experiment, and p, which is the probability of success in a single trial. An important problem in statistics is to estimate the population parameters, given a random sample drawn from the population. If the form of the frequency function is assumed, then values for its parameters may be determined by forming the likelihood function, which gives the probability that the given sample came from a population

with the assumed frequency function. Hence, the likelihood function is a function of the unknown parameters. The values of the parameters are now estimated by maximizing the likelihood function with respect to these parameters, subject to any constraints that may be present. The resulting optimal values the parameters are known as maximum likelihood estimates. The method may be applied to functions of discrete or continuous variables.

1.2.2 Aerodynamics [120]

There are many optimization problems concerned with the design, performance and flying qualities of aircraft. The aircraft designer must minimize the structural weight, subject to the structure having sufficient strength and stiffness to carry the critical design loads safely. The cruising altitude should be chosen so as to minimize fuel consumption; it often happens that a steady climb is more economical than flight at constant altitude. Aircraft are designed for many different purpose, and in particular cases, it may be important to the following problems:

(i) minimize the take-off run;

(ii) maximum the rate of climb;

(iii) maximize the ceiling;

(iv) maximize the endurance;

(v) minimize the wave drag in supersonic flight.

All these problems are subject to various constraints which, in certain cases, may be so severe that no optimization problem remains.

1.2.3 Factory Allocation Problem [37]

The manager of a factory manufacturing different items of equipment, each with a different profit margin, employs in his factory a certain number of men, and possesses between the different manufactured items so as to make most profit, and also how to arrange the processes inside the factory so that the production proceeds smoothly and no predictable hold-up occurs, giving rise to an unnecessary loss in profits. He is also probably interested to know whether it would be worth his while to employ more men and the other equipment.

1.2.4 Parameter Problem [37]

A physical experiment involving the measurement of an independent and several dependent variables is carried out a large number of times. The relationship between these variables is known theoretically, but involves several unknown parameters. Those values of the parameters for which the theoretical relationship between the variables is as close as possible to the experimental results are to be found.

1.2.5 Chemical Engineering [120]

The manager of a chemical plant has to decide on his major objective in running the plant. Should he maximize output? Is this consistent with maximizing profit? To answer these questions requires the solution of at least two optimization problems. The answer to the second question may be 'No', for lower output could mean better quality output, greater efficiency and more variable by-products.

1.2.6 Operational Research [120]

The application of optimization techniques to industrial and commercial problems forms part of the subject of operational research. The fundamental problem of stock control is to choose a stock level and a stock replacement policy which maximize overall profit. The usual assumptions are that losses are incurred if either too much or too little stock is kept. The demand may be known exactly or its frequency function may be assumed. A related problem is that of renewing obsolescent machinery while maintaining maximum efficiency.

1.2.7 Economics [120]

How many new power stations should be built in Hong Kong or mainland China between now and the year 2020? How many of them should be atomic power stations? These questions lead to very complicated optimization problems; it is not at all clear which quantities should be maximized or minimized, and it is even less clear what constraints should be imposed. Nevertheless, problems of this kind obviously need careful study before the crucial decisions are taken.

1.3 Mathematical Models for Optimization Problems

The first essential step in any optimization problem is to decide on the physical variable that is to be optimized, that is, the entity that is to attain its 'least' or 'greatest' value at the solution of the problem. This concept will differ in the various applications and could be either monetary cost, time taken, fuel used,

discomforts created, error between achievement and target, or some other function. Having decided on this function, which will in general be called the cost function, it must then be expressed in terms of the variables of the variables of the problems. These variables can be formed into a vector $x \in \mathbb{R}^n$, which defines the space in which the problem is posed.

Suppose we wish to minimize a real-valued cost function f, where $f : \mathbb{R}^n \to \mathbb{R}$, of n variables.

Definition. *A local minimizer of $f : \mathbb{R}^n \to \mathbb{R}$ is a point x^* such that*

$$f(x^*) \leq f(x) \tag{1.1}$$

for all x near x^*.

It is standard to express this problem as

$$\min_x f(x) \tag{1.2}$$

or to say that we seek to solve the problem $\min f$. The understanding is that (1.1) means that we seek a local minimizer. We will refer to f as the cost function and to $f(x^*)$ as the minimum or minimum value. If a local minimizer x^* exists, we say a minimum is attained at x^*.

We say that the problem (1.2) is unconstrained optimization problem since we impose no conditions on the independent variables x and assume that f is defined for all $x \in \mathbb{R}^n$.

The local minimization problem is different and much easier than the global minimization problem in which a global minimizer, a point $x^* \in \mathbb{R}^n$ such that

$$f(x^*) \leq f(x) \tag{1.3}$$

for all x is sought.

The constrained optimization problem is to minimize function f over a set $D \subset \mathbb{R}^n$. Therefore, a local minimizer is an $x^* \in D$ such that

$$f(x^*) \leq f(x) \tag{1.4}$$

for all $x \in D$ near x^*. Similar to (1.2), we express this as

$$\min_{x \in D} f(x) \tag{1.5}$$

or say that we seek to solve the problem $\min_D f$. A global minimizer is a point $x^* \in D$ such that

$$f(x^*) \leq f(x) \tag{1.6}$$

for all $x \in D$. For more details about constrained optimization and pointers to software, please refer to the books [48], [57], [85] and [78].

1.4 Unconstrained Optimization Techniques

This thesis is concerned with unconstrained optimization, that is, with minimizing (or maximizing) a function, say $f(x)$, where $f : \mathbb{R}^n \to \mathbb{R}$, which is not subject to any constraints. The function may be one of continuous variables or discrete variables, or a mixture of the two.

1.4.1 Direct Method - Differential Calculus

We will start by considering the unconstrained optimization of functions of continuous variables. Now, you may well imagine that all these problems can be

solved using the differential calculus. To minimize a function f of n variables, say $f(x_1, ..., x_n)$, we simply have to satisfy the n simultaneous equations:

$$\frac{\partial f}{\partial x_j}(x_1, ..., x_n) = 0, \quad \text{for } j = 1, ..., n. \tag{1.7}$$

The calculus approach is useful if the equations can be solved directly, for example, if they are all linear. It is also useful if it enables the dimension of the problem to be reduced. Let us consider the following example [11]:

Let f be a quadratic function in x_1 for fixed values of the other variables. Then, we can write

$$f(x_1, x_2, ..., x_n) = g_0(x_2, ..., x_n) + g_1(x_2, ..., x_n)x_1 + g_2(x_2, ..., x_n)x_1^2.$$

So, this implies that

$$\frac{\partial f}{\partial x_1} = g_1(x_2, ..., x_n) + 2g_2(x_2, ..., x_n)x_1.$$

Note that if $g_2 < 0$ for any values of $x_2, ..., x_n$, then we can make the function f arbitrarily large and negative by making x_1 large enough. Also, if $g_2 = 0$ while $g_1 \neq 0$, then we can make f again arbitrarily large and negative. Otherwise, we can find that

$$\min_{x_1} f(x) = g_0(x_2, ..., x_n) - \frac{g_1(x_2, ..., x_n)^2}{4g_2(x_2, ..., x_n)}.$$

We can now have an optimization problem in only $n - 1$ variables. When this is solved, we can derive the corresponding value of x_1 from the following formula:

$$x_1 = -\frac{g_1(x_2, ..., x_n)}{2g_2(x_2, ..., x_n)}.$$

However, the differential calculus method just provides the equations, but does not provide a method for solving such equations if they have no exploitable special structure. On the other hand, if the functions $f_j(x)$ are all real-valued, for all $j = 1, ..., n$, we can solve the equations:

$$f_j(x) = 0$$

by minimizing the sum of the squares of the residuals defined by

$$f(x) := \sum_{j=1}^{n} f_j^2(x). \tag{1.8}$$

We have apparently gone full circle: to minimize the function $f(x)$, we need to solve a set of equations which may be solved by minimizing another function. The question arises whether either of these transformations is of any value.

1.4.2 Iterative Methods

If the direct methods fail, we may use iterative methods. These are particularly convenient with a computer, because once a single iteration has been performed, we need very little extra programming to do an arbitrarily large number of iterations.

Iterative methods are often useful for solving simultaneous equations once, they have been transformed into a minimization problem such as the equations (1.8). Unfortunately, they are not so useful for solving the calculus equations (1.7), primarily because stationary points are not necessarily minima and we may iterate towards a saddle point, or even a maximum.

It is better to iterate directly on the function f and use a "valley-descending" method. "Valley-descending" means finding a minimum of a function $f(x)$ using the following strategy [11]:

(1) Given an initial guess, say $x_0 \in \mathbb{R}^n$;

(2) For $k = 0, 1, ...$, take a trail solution, say x_k;

(3) Find a direction from this trail solution in which $f(x)$ decreases;

(4) Find a point x_{k+1} in this direction such that $f(x_{k+1}) < f(x_k)$;

(5) Repeat the process from this new trial solution.

If the function $f(x)$ is to be maximized, the corresponding strategy where we require $f(x_{k+1})$ to be greater than $f(x_k)$ can be called "hill-climbing". Such a strategy can easily be implemented on a computer. However, when using iterative methods on a computer, we must beware of methods that work well on some problems, but go into endless loops or just fail on others.

1.5 Main Objectives of the Thesis

There has been an immense amount of work in the past on numerical methods for unconstrained optimization of functions of n variables where $n \geq 1$. Most of this has been concerned with finding local optima of functions that are twice-differentiable, so it will be concentrated on this.

Most people might reasonably hope that the work in this area has led to at least one method which is effective for all problems of this type, but unfortunately, it will never be possible, since the choice of method must depend on the answers to the following questions [11]:

(1) How easy is it to calculate function values?

(2) Can we easily calculate the gradient vector at any trial solution; in other words, can we easily calculate all first derivatives?

(3) Can we easily calculate the Hessian matrix at any trial solution; in other words, can we easily calculate all second derivatives?

(4) Can we afford to store an $n \times n$ matrix representing the true or estimated Hessian, or alternatively, its inverse?

(5) Is the Hessian matrix sparse, that is, are many of its elements zero? If so, should we take advantage of this fact?

(6) Does the problem have any special features that we should exploit to make the solution easier to find?

Since there are no easy answers to such questions industry and commerce will always require specialized help to solve practical problems of this nature. The choice of method to solve each problem is likely to be better if the strengths and weaknesses of the more successful existing methods are understood.

In this thesis, I will discuss four existing methods which make use of quite different techniques and have different advantages and limitations. The first, *steepest descent method*, which is discussed in Chapter 3, requires only the calculation of the first derivatives and the computation is simple. But the convergence is usually very slow. The second, *Newton's method*, requires the calculation of the second derivatives and can be modified to incorporate line searches to improve its reliability, but it costs a large volume of computation. So, I will discuss some other cheaper iterative methods using the second derivative in Chapter 4.

Two other methods calculate only the first, not the second derivatives, but nevertheless converge in a finite number of steps when the function f is quadratic. The first of these, known as the *conjugate gradient method*, which is a main type of the *multi-step methods* (see Chapter 5), has the advantage for large scale problems that it does not require the storage of any large $n \times n$ matrices. The other method, known as a *quasi-Newton method* (see Chapter 6), builds up an explicit approximation to the Hessian matrix and is generally more effective than conju-

gate gradients when the computer has sufficient storage.

Chapter 2

Basic Concepts in Optimizations of Smooth Functions

The primary goal of this chapter is to give the general theoretical background necessary to understand the remaining chapters. Also, a few topics of general nature are discussed.

2.1 Notation

In this thesis, following the convention [75], vectors are to be understood as column vectors. The vector x^* will denote a solution, x a potential solution, $\{x_k\}_{k\geq 0}$ the sequence of iterates. We will refer to x_0 as the initial iterate. x_0 is sometimes timidly called the initial guess. We will denote the ith component of a vector x by $(x)_i$ and the ith component of x_k by $(x_k)_i$. we will rarely need to refer to individual components of vectors. We will let $\frac{\partial f}{\partial x_i}$ denote the partial derivative of f with respect to $(x)_i$. $e = x - x^*$ will denote the error, $e_n = x_n - x^*$ the error

in the nth iterate, and $B(r)$ the ball of radius r about x^*

$$B(r) = \{x : \|x - x^*\| < r\}.$$

For $x \in \mathbb{R}^n$, we let $\nabla f(x) \in \mathbb{R}^n$ denote the gradient of f,

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}, ..., \frac{\partial f}{\partial x_n}\right)^T,$$

when it exists.

We let $\nabla^2 f$ denote the Hessian of f:

$$(\nabla^2 f)_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j},$$

when it exists. Note that $\nabla^2 f$ is the Jacobian of ∇f. However, $\nabla^2 f$ has more structure than a Jacobian for a general non-linear function. If f is twice continuously differentiable, then the Hessian is symmetric, that is, $(\nabla^2 f)_{ij} = (\nabla^2 f)_{ji}$ for all $i, j = 1, ..., n$, by equality of mixed partial derivatives [106].

In this thesis, we will consistently use the Euclidean norm

$$\|x\| = \sqrt{\sum_{i=1}^{n} (x)_i^2}.$$

When we refer to a matrix norm, we will mean the matrix norm induced by the Euclidean norm

$$\|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|}.$$

In optimization definiteness or semi-definiteness of the Hessian plays an important role in the necessary and sufficient conditions for optimality and in our choice of algorithms throughout this thesis.

Definition. *An $n \times n$ matrix A is positive semi-definite if $x^T Ax \geq 0$ for all $x \in \mathbb{R}^n$. A is positive definite if $x^T Ax > 0$ for all $x \in \mathbb{R}^n$, $x \neq 0$. If A has both positive and negative eigenvalues, we say A is indefinite.*

We will use two forms of the fundamental theorem of calculus [75], one for the function-gradient pair and one for the gradient-Hessian.

Theorem 2.1 *Let f be twice continuously differentiable in a neighborhood of a line segment between points $x^*, x = x^* + e \in \mathbb{R}^n$, then*

$$f(x) = f(x^*) + \int_0^1 \nabla f(x^* + te)edt \tag{2.1}$$

and

$$\nabla f(x) = \nabla f(x^*) + \int_0^1 \nabla^2 f(x^* + te)edt. \tag{2.2}$$

A direct consequence of Theorem 2.1 is the following form of Taylor's Theorem, we will use throughout this thesis.

Theorem 2.2 *Let f be twice continuously differentiable in a neighborhood of a point $x^* \in \mathbb{R}^n$. Then, for $e \in \mathbb{R}^n$ and $\|e\|$ sufficiently small*

$$f(x^* + e) = f(x^*) + \nabla f(x^*)^T e + \frac{1}{2} e^T \nabla^2 f(x^*) e + o(\|e\|^2). \tag{2.3}$$

2.2 Different Types of Minimizer

Although any function $f(x)$ must have a least value, the value is not necessarily finite. It could even be that $f(x)$ does not take its least value in $\mathbb{R}^n$. A simple example of this is when $f(x)$ is a linear function other than a constant. What most people are interested in when presented with the minimization problem is a solution x^* of a character and this leads us to the following definitions [86]:

Definition. *A point x^* is said to be a strong local minimizer of $f(x)$ if $f(x)$ is defined on a δ-neighbourhood of x^* and there exists an ε, where $0 < \varepsilon < \delta$, such*

that

$$f(x^*) < f(x), \tag{2.4}$$

for all points such that $0 < \|x^* - x\| < \varepsilon$.

Definition. *Let* $f(x)$ *be defined on a* δ*-neighbourhood of* x^*. *The function* $f(x)$ *is said to have a weak local minimizer at* x^* *if* x^* *is not a strong local minimum, but there exists an* ε, *where* $0 < \varepsilon < \delta$, *such that*

$$f(x^*) \leq f(x), \tag{2.5}$$

for all points such that $0 < \|x^* - x\| < \varepsilon$.

Definition. *A point* x^* *is said to be a global minimizer of a function* $f(x)$ *if*

$$f(x^*) \leq f(x), \quad \forall x \in \mathbb{R}^n. \tag{2.6}$$

For an arbitrary function, there is no guarantee that such an x^* exists since $f(x)$ may take its least value at a limit as $\|x\| \to \infty$.

The figure 2.1 illustrates the different types of minima.

When it is not necessary to distinguish between a strong and weak local minimum only the term local minimum will be used.

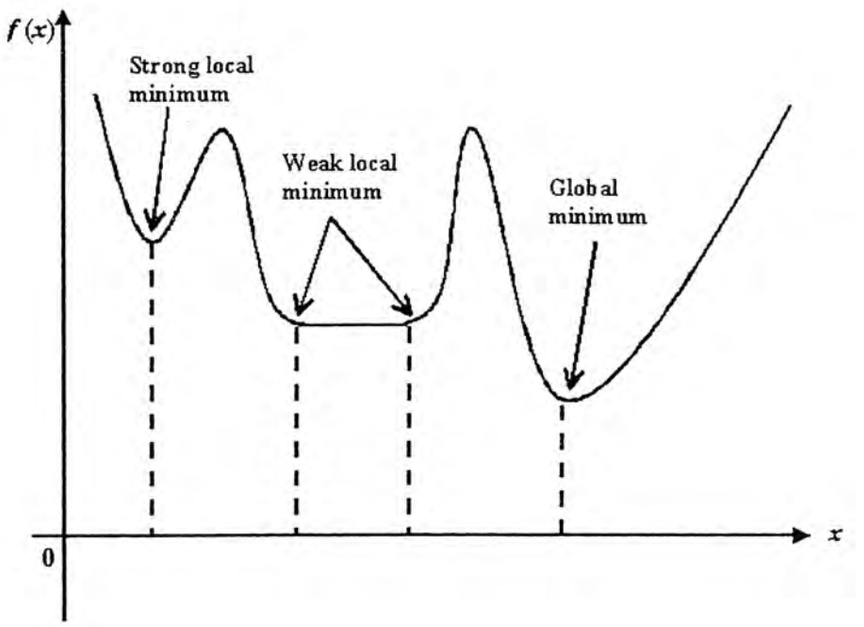

Figure 2.1: A function $f(x)$ of single variable.

2.3 Necessary and Sufficient Conditions for Optimality

Nothing constructive can be said about the behavior of $f(x)$ at x^* unless $f(x)$ has certain continuity properties.

Consider the following example with a discontinuous function [86]. In particular, one such that

$$\lim_{\alpha \to 0} [f(x^*) - f(x^* + \alpha d)] \neq 0,$$

where d is an arbitrary unit vector, and α is a scalar.

An example of such a function is illustrated in Figure 2.2. The function is

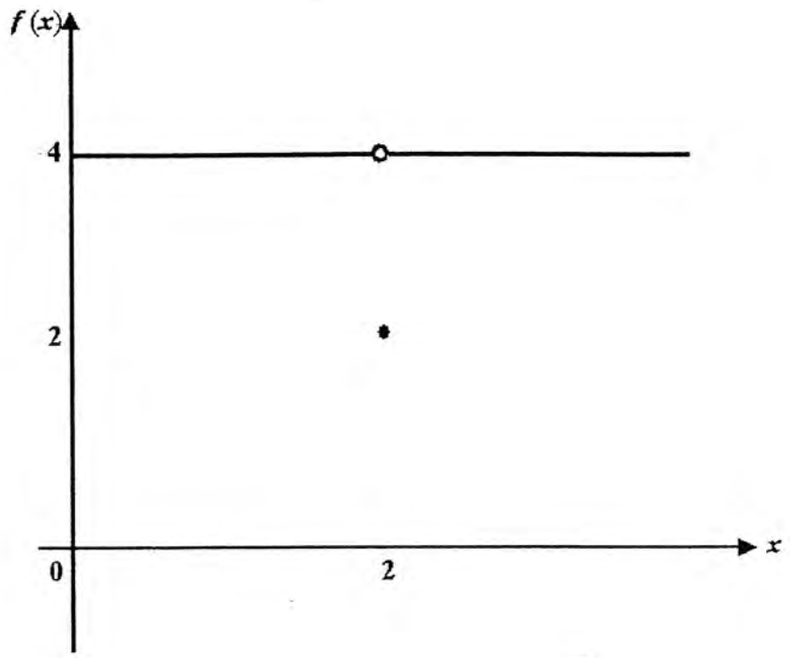

Figure 2.2: An example with a discontinuous function $f(x)$.

that:

$$f(x) = \begin{cases} 4, & \text{if } x \neq 2, \\ 2, & \text{if } x = 2. \end{cases}$$

It is unlikely that an algorithm which is efficient on well-behaved functions would find the minimum at $x = 2$. Consequently, algorithms are designed to find local minima of only a specific class of functions. Usually, an algorithm will utilize properties of the class of functions to be minimized. Most algorithms described in this thesis assume that

$$f(x) \in C^2, \quad \text{for all } x \in \mathbb{R}^n. \tag{2.7}$$

Now, let f be twice continuously differentiable. We will use Taylor's theorem in a simple way to show that the gradient of f vanishes at a local minimizer and the Hessian matrix of f is positive semi-definite. These are the necessary conditions for optimality [75].

The necessary conditions relate (1.2) to a non-linear equation and allow one to use fast algorithms for non-linear equations [35], [74] and [89] to compute minimizers. A critical first step in the design of an algorithm for a new optimization problem is the formulation of necessary conditions. Of course, the gradient vanishes also at a maximum, therefore the utility of the nonlinear equations formulation must be restricted to a neighborhood of a minimizer.

Theorem 2.3 *Let f be twice continuously differentiable and let x^* be a local minimizer of f. Then,*

$$\nabla f(x^*) = 0. \tag{2.8}$$

Moreover, $\nabla^2 f(x^*)$ is positive semi-definite.

Proof: Let $u \in \mathbb{R}^n$ be given. By Taylor's Theorem, it states that for all real scalar t sufficiently small

$$f(x^* + tu) = f(x^*) + t\nabla f(x^*)^T u + \frac{t^2}{2} u^T \nabla^2 f(x^*) u + o(t^2).$$

Since x^* is a local minimizer, then we must have for t sufficiently small,

$$f(x^* + tu) - f(x^*) \geq 0$$

and

$$\nabla f(x^*)^T u + \frac{t}{2} u^T \nabla^2 f(x^*) u + o(t) \geq 0 \tag{2.9}$$

for all t sufficiently small and all $u \in \mathbb{R}^n$. So, if we set $t = 0$ and $u = -\nabla f(x^*)$, we obtain

$$\|\nabla f(x^*)\|^2 = 0.$$

Setting $\nabla f(x^*) = 0$, dividing by t and setting $t = 0$ in (2.9), we have:

$$\frac{1}{2} u^T \nabla^2 f(x^*) u \geq 0$$

for all $u \in \mathbb{R}^n$. □

This condition $\nabla f(x^*) = 0$ is called the first-order necessary condition and a point satisfying that condition is called a stationary point or a critical point.

However, a stationary point need not be a minimizer. For example, the function $f(x) = -x^{2n}$, where $n \geq 2$, satisfies the necessary conditions at $x = 0$, which is a maximizer of f. To obtain a minimizer, we must require that the second derivative be non-negative. This alone is not sufficient and only if the second derivatives is strictly positive can we be completely certain. These are the sufficient conditions for optimality [75].

Theorem 2.4 *Let f be twice continuously differentiable in a neighborhood of x^*. Assume that $\nabla f(x^*) = 0$ and that $\nabla^2 f(x^*)$ is positive definite. Then, x^* is a local minimizer of f.*

Proof: Let $u \in \mathbb{R}^n$ such that $u \neq 0$. For sufficiently small t, we have:

$$\begin{aligned} f(x^* + tu) &= f(x^*) + t\nabla f(x^*)^T u + \frac{t^2}{2} u^T \nabla^2 f(x^*) u + o(t^2) \\ &= f(x^*) + \frac{t^2}{2} u^T \nabla^2 f(x^*) u + o(t^2). \end{aligned}$$

Hence, if $\lambda > 0$ is the smallest eigenvalue of $\nabla^2 f(x^*)$, we have:

$$f(x^* + tu) - f(x^*) \geq \frac{\lambda}{2} \|tu\|^2 + o(t^2) > 0$$

for t sufficiently small. Hence, x^* is a local minimizer. □

From the above two theorems, it follows that the Hessian matrix $G := \nabla^2 f$ of f must be at least positive semi-definite for x^* to be a local minimizer and that for G positive definite, x^* is a strong local minimizer. For the functions with higher continuous derivatives and G positive semi-definite, it is possible to

give additional conditions to distinguish between the different type of stationary points. Since it is impractical to verify these conditions computationally, I do not pursue the matter further. For details of these higher-order conditions, please refer to Gue and Thomas [64] for deeper discussions.

2.4 Quadratic Functions

It is unlikely that a minimization algorithm would be required to find the minimum of a quadratic function, since it will be shown that this is equivalent to solving a set of linear simultaneous algebraic equations. regarding of this, the behavior of an algorithm on a quadratic function is important in optimization. One reason for this is that it is the simplest type of function that can be minimized since all linear functions apart from $f(x)$ =constant are unbounded below. A more important reason is that the behavior of a minimization algorithm on a quadratic function is indicative of its behavior in the neighbourhood of the solution when $f(x) \in C^2$.

Consider the following quadratic function:

$$f(x) = \frac{1}{2}x^T Ax - b^T x + c, \tag{2.10}$$

where A is an $n \times n$ matrix, b is an $n \times 1$ vector and c is a scalar. Both A and b have constant elements.

Without loss of generality, we may assume that A is symmetric because

$$x^T Ax = x^T \left(\frac{A + A^T}{2} \right) x. \tag{2.11}$$

Then, it is clear that

$$G(x) := \nabla^2 f(x) = A$$

for all x. The symmetry of A implies that

$$g(x) := \nabla f(x) = Ax - b.$$

From the necessary conditions for a minimum given in Theorem 2.3, it follows that

$$Ax - b = 0.$$

Although $f(x)$ may not posses a local minimum, it must posses a stationary point if b lies in the range of A. When A is non-singular, there is a unique stationary point. If A is positive definite, then

$$x^* = A^{-1}b,$$

is a strong local minimizer. If A is positive semi-definite, then $f(x)$ has a weak local minimum if b lies in the range of A.

When $f(x)$ has a weak local minimum say at x^*, then

$$y = x^* + z,$$

where z is any vector lying in the null space of A, is also a weak local minimum.

An important concept [86] when minimizing a quadratic function is that of conjugacy. It can be shown that there exists an $n \times n$ matrix P of rank n such that

$$P^T AP = D,$$

where D is a diagonal matrix. There are many such matrices, for example, the columns of P can be the eigenvectors of A. If we make the transformation of variables

$$x = Py,$$

then

$$f(y) = \frac{1}{2}y^T Dy - b^T Py + c,$$

so that $f(y)$ is a separable function of the variables $(y_1, ..., y_n)$. The minimum of a separable function can be found by minimizing with respect to the variables individually. This is identical to minimizing $f(x)$ along the vectors $p_1, p_2, ..., p_n$, where p_j is the jth column of P. The directions $p_1, p_2, .., p_n$ are called a conjugate set of directions. For more discussion, please refer to [74], [3], [8] and [60].

On the other hand, if A is indefinite, then the necessary conditions in Theorem 2.3 imply that there will be no local minimum [75]. But it is still important to understand some properties of quadratic problems with indefinite Hessian matrix A when we design algorithms with initial iterates far from local minimizers.

If we have:

$$u^T Au < 0,$$

for $u \in \mathbb{R}^n$, then we say that u is a direction of negative curvature. If u is a direction of negative curvature, then $f(x+tu)$ will be decreased to $-\infty$ as $t \to \infty$.

2.5 Convex Functions

It is not usually known or possible to determine whether a function is convex. Despite this, the convex functions and their properties are important in optimization. The reason for this is that convergence of a minimization algorithm can often be proved for convex functions. Since the convex functions are so rare, this would not be particularly valuable were it not for the fact that many functions are convex in the neighbourhood of a local minimum [86].

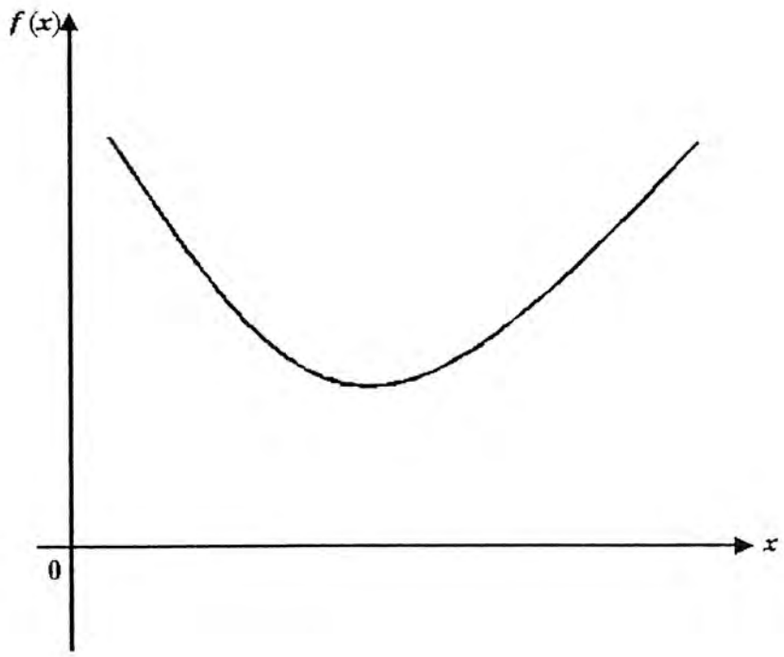

Figure 2.3: A strictly convex function.

Definition. *A function $f(x)$ is said to be strictly convex if for any two points $x, y \in \mathbb{R}^n$, we have:*

$$f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda)f(y), \tag{2.12}$$

for any $0 < \lambda < 1$.

Definition. *A function $f(x)$ is said to be convex if for any two points $x, y \in \mathbb{R}^n$, we have:*

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y), \tag{2.13}$$

for any $0 < \lambda < 1$.

The difference between a strictly convex function and a convex function is illustrated in Figures 2.3 and 2.4.

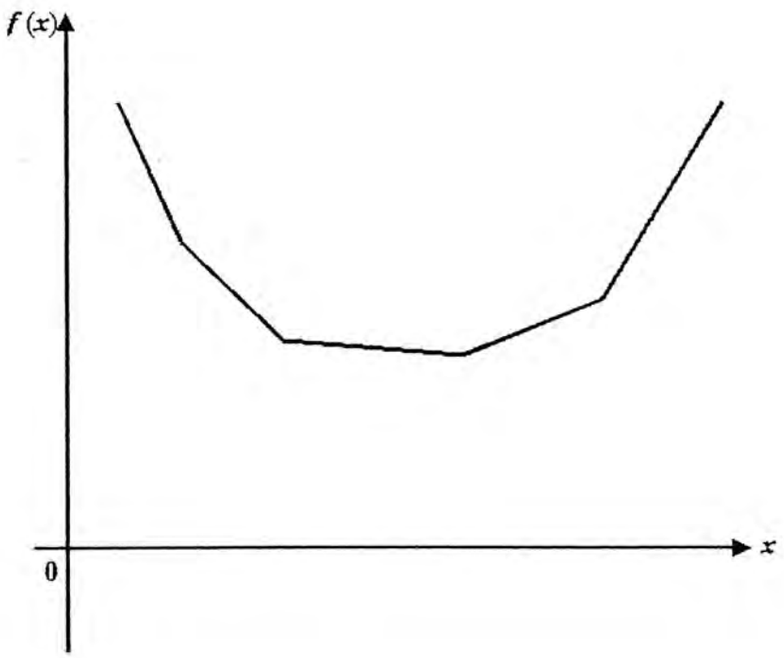

Figure 2.4: A convex function.

If $f(x) \in C^2$ and is strictly convex, then the Hessian matrix G is positive semi-definite at every point in $\mathbb{R}^n$. Furthermore, if $f(x)$ is a quadratic function and G is positive definite, then $f(x)$ is strictly convex. If $f(x)$ is a convex function and has a strong local minimum, then this is unique and is the global minimum. Should $f(x)$ have a weak local minimum and x^* and y^* are two such minima, then

$$f(x^*) = f(y^*)$$

and all the points on the line joining x^* and y^* are weak local minima.

It is important to have analytic criteria to evaluate whether a function is convex or not. Such criteria exist and are simplest for differentiable functions. They are based on the following properties ([128], [31], [110], [111], [127], [68]):

Theorem 2.5 *Suppose $f(x)$ is a differentiable function in $\mathbb{R}^n$. Then, $f(x)$ is a*

convex function in $\mathbb{R}^n$ *if and only if*

$$f(y) \geq f(x) + \nabla f(x)^T(y - x) \tag{2.14}$$

for all $x, y \in \mathbb{R}^n$.

Proof:

"Only if" part: Suppose $f(x)$ is a convex function. Then, for all $0 < \lambda < 1$, we have:

$$\begin{aligned} f(\lambda y + (1-\lambda)x) &\leq \lambda f(y) + (1-\lambda) f(x) \\ \frac{f(x + \lambda(y-x)) - f(x)}{\lambda} &\leq f(y) - f(x). \end{aligned}$$

Let $\lambda \to 0$, then

$$\nabla f(x)^T(y - x) \leq f(y) - f(x).$$

"If" part: Suppose the statement (2.14) holds. Take any $x_1, x_2 \in \mathbb{R}^n$ and $0 < \lambda < 1$. Let $x = \lambda x_1 + (1-\lambda)x_2 \in \mathbb{R}^n$, then we have:

$$\begin{aligned} f(x_1) &\geq f(x) + \nabla f(x)^T(x_1 - x), \\ f(x_2) &\geq f(x) + \nabla f(x)^T(x_2 - x). \end{aligned}$$

So,

$$\begin{aligned} \lambda f(x_1) + (1-\lambda) f(x_2) &\geq f(x) + \nabla f(x)^T[\lambda x_1 + (1-\lambda)x_2 - x] \\ &= f(\lambda x_1 + (1-\lambda)x_2). \end{aligned}$$

Hence, $f(x)$ is a convex function. □

Theorem 2.6 *Suppose* $f(x)$ *is a twice continuously differentiable function in* $\mathbb{R}^n$. *Then,* $f(x)$ *is a convex function in* $\mathbb{R}^n$ *if and only if the Hessian matrix* $G(x) = \nabla^2 f(x)$ *is at least positive semi-definite for all* $x \in \mathbb{R}^n$.

Proof:

"If" part: Suppose the Hessian matrix $\nabla^2 f(x)$ is at least positive semi-definite for all $x \in \mathbb{R}^n$. By Mid-value Theorem, for any x, $\bar{x} \in \mathbb{R}^n$, we have:

$$f(x) = f(\bar{x}) + \nabla f(\bar{x})^T(x - \bar{x}) + \frac{1}{2}(x - \bar{x})^T \nabla^2 f(\hat{x})(x - \bar{x}),$$

where $\hat{x} = \bar{x} + \theta(x - \bar{x})$, $\theta \in (0, 1)$. Note that $\hat{x} \in \mathbb{R}^n$, then by assumption, we have:

$$f(x) \geq f(\bar{x}) + \nabla f(\bar{x})^T(x - \bar{x}).$$

By Theorem 2.5, f is a convex function.

"Only if" part: Suppose f is a convex function. Take any $\bar{x} \in \mathbb{R}^n$, we need to prove that $d^T \nabla^2 f(\bar{x}) d \geq 0$, $\forall d \in \mathbb{R}^n$.

For any α such that $|\alpha| < \delta$, where $\delta > 0$, then $\bar{x} + \alpha d \in \mathbb{R}^n$. By Theorem 2.5, then

$$f(\bar{x} + \alpha d) \geq f(\bar{x}) + \alpha \nabla f(\bar{x})^T d. \tag{2.15}$$

Also, since $f(x)$ is twice continuously differentiable, then

$$f(\bar{x} + \alpha d) = f(\bar{x}) + \alpha \nabla f(\bar{x})^T d + \frac{\alpha^2}{2} d^T \nabla^2 f(\bar{x}) d + o(\|\alpha d\|^2). \tag{2.16}$$

From (2.15) and (2.16), we have:

$$\frac{1}{2}\alpha^2 d^T \nabla^2 f(\bar{x}) d + o(\|\alpha d\|^2) \geq 0$$

Dividing both sides by α^2 and let $\alpha \to 0$, then

$$d^T \nabla^2 f(\bar{x}) d \geq 0.$$

Hence, $\nabla^2 f(x)$ is at least positive semi-definite. □

2.6 Existence, Uniqueness and Stability of a Minimum

The problems of existence, uniqueness and stability of a solution are an important part of the mathematical theory for minimization problems. In particular, it is very important for the problems of the unconstrained optimization. Following [94], we discuss these problems in this subsection.

2.6.1 Existence of a Minimum

The question of the existence of a minimum point is usually solved quite simply by means of the following theorem:

Theorem 2.7 *Let $f(x)$ be a continuous function on $\mathbb{R}^n$ and let the set $Q_\alpha = \{x : f(x) \leq \alpha\}$, for some $\alpha \in \mathbb{R}$, be non-empty and bounded. Then, there exists a global minimum point of $f(x)$ on $\mathbb{R}^n$.*

Proof: Let

$$f(x_k) \to \inf_{x \in \mathbb{R}^n} f(x) < \alpha.$$

Then, $x_k \in Q_\alpha$ for sufficiently large k. Since $f(x)$ is a continuous function, then the set Q_α is closed. Therefore, the set Q_α is compact since it is closed and bounded. Hence, the sequence x_k has a limit point $x^* \in Q_\alpha$. It follows from the continuity of $f(x)$ that

$$f(x^*) = \inf_{x \in \mathbb{R}^n} f(x),$$

that is,

$$x^* = \min_{x \in \mathbb{R}^n} f(x).$$

Hence, there exists a global minimum x^* of $f(x)$. □

Note that the assumption of the boundedness of Q_α is essential. In some special cases, we can also prove the existence of a solution in the situations which is not covered by Theorem 2.7.

2.6.2 Uniqueness of a Minimum

Definition. *We say that a minimum of a function $f(x)$ is locally unique if in some neighborhood of this minimum point, there are no other minimizers.*

Definition. *We say that x^* is a non-singular minimizer if at the x^*, the second-order sufficient condition holds, that is,*

$$\nabla f(x^*) = 0 \quad \textit{and} \quad \nabla^2 f(x^*) > 0. \tag{2.17}$$

Theorem 2.8 *A non-singular minimum is locally unique.*

Proof: By Taylor's Theorem, we have:

$$\nabla f(x) = \nabla f(x^*) + \nabla^2 f(x^*)(x - x^*) + o(x - x^*). \tag{2.18}$$

Let $\lambda > 0$ be the smallest eigenvalue of $\nabla^2 f(x^*)$. Since $\nabla^2 f(x^*) > 0$, then $\|\nabla^2 f(x^*)x\| \geq \lambda \|x\|$ for all x. Hence, we have:

$$\begin{aligned} \|\nabla f(x)\| &= \|\nabla^2 f(x^*)(x - x^*)\| + o(\|x - x^*\|) \\ &\geq \lambda \|x - x^*\| + o(\|x - x^*\|) \\ &> 0 \end{aligned}$$

for sufficiently small $\|x - x^*\|$. Therefore, in some neighborhood of x^*, there are no stationary points of $f(x)$. Hence, there are also no minimum points. □

For convex functions, the answer to the question of uniqueness of a minimum is easy to obtain.

Theorem 2.9 *A minimum of a strictly convex function is (globally) unique.*

Proof: The proof follows immediately from Theorems 2.6 and 2.8. □

2.6.3 Stability of a Minimum

In practical solution of optimization problems, one is continually faced with the following question. Suppose we have discovered a method for constructing a minimizing sequence. Does it converge to the solution? If, instead of the initial minimization problem, can one assert that the solutions are close? Questions like these are the province of optimization theory and involve the notions of stability and correctness. We will use the term "stability" for optimization problems and leave the term "correctness" for problems not involving optimization.

Definition. *The local minimum x^* of $f(x)$ is called locally stable if every local minimizing sequence converges to it. That is, there is a $\delta > 0$ such that $f(x_k) \to f(x^*)$, $\|x_k - x^*\| \leq \delta$, then we have:*

$$x_k \to x^*.$$

Theorem 2.10 *A local minimum of a continuous function $f(x)$ is locally stable if it is locally unique.*

Proof: Let x^* be locally unique. Take an arbitrary local minimizing sequence x_k, $\|x_k - x^*\| \leq \delta$, $f(x_k) \to f(x^*)$. By the compactness of a unit sphere in $\mathbb{R}^n$, one can take a convergent subsequence $x_{k_i} \to \bar{x}$, $\|\bar{x} - x^*\| \leq \delta$. It follows from the continuity of $f(x)$ that $f(\bar{x}) = \lim f(x_{k_i}) = f(x^*)$. Then, however, $\bar{x} = x^*$ since x^* is a locally unique minimum point. Since the same is true for any other sequence, the entire sequence x_k converges to x^*. Hence, x^* is locally stable. □

Theorem 2.11 *Let x^* be a locally stable minimum of the continuous function $f(x)$ and let $g(x)$ be a continuous function. Then, for sufficiently small $\varepsilon > 0$, the function $f(x) + \varepsilon g(x)$ has a local minimum x_ε in a neighborhood of x^* and $x_\varepsilon \to x^*$ as $\varepsilon \to 0$.*

Therefore, the stability property implies that the minimum of the initial function and that of the perturbed function are close.

A non-singular minimum, as follows from Theorems 2.8 and 2.10, is locally stable. In this case, the result of Theorem 2.11 can be refined.

Theorem 2.12 *Let x^* be a non-singular minimum of $f(x)$ and let a function $g(x)$ be continuously differentiable in a neighborhood of x^*. Then, for sufficiently small $\varepsilon > 0$, there exists a local minimum x_ε of the function $f(x) + \varepsilon g(x)$ in a neighborhood of x^*, and*

$$x_\varepsilon = x^* - \varepsilon \left[\nabla^2 f(x^*)\right]^{-1} \nabla g(x^*) + o(\varepsilon).$$

We can also consider the notion of the global stability of minimum points. This can just be done by replacing the word "local" by the word "global" in the definition. Namely, a global minimum is said to be globally stable if any

minimizing sequence converges to it. In this case, we speak of global stability of the minimization problem. Repeating almost verbatim the proof of Theorem 2.10, we obtain that if x^* is the unique global minimum of the continuous function $f(x)$ and the set $Q_\alpha = \{x : f(x) \leq \alpha\}$ is nonempty and bounded for some $\alpha > f(x^*)$, then x^* is global stable. The requirement for the set Q_α to be bounded is essential, for example, for the function

$$f(x) = \frac{x^2}{1+x^4}, \quad x \in \mathbb{R},$$

the global minimum $x^* = 0$ is unique, but not globally stable, since the minimizing sequence $x_k \to \infty$ does not converge to x^*.

We can consider the following broader definition of stability which does not include uniqueness of a minimum:

Definition. *The set X^* of global minimum points of $f(x)$ is said to be weakly stable if all limit points of any minimizing sequence belong to X^*.*

And we have the criterion for weak stability is given as follows:

Theorem 2.13 *Suppose $f(x)$ is continuous and the set $Q_\alpha = \{x : f(x) \leq \alpha\}$ is nonempty and bounded for some $\alpha > \inf f(x)$. Then, the set of minimum points of $f(x)$ is weakly stable.*

For more details for the general well-posed mathematical problems, please refer to Ivanov, Vasin and Tanana [71], and Tikhonov and Arsenin [116]. For optimization problems, please refer to Vasil'ev [118], Karmanov [73], Bank [4] and Fiacco [40].

2.7 Types of Convergence

By a local convergence method, we mean one that requires that the initial iterate x_0 is close to a local minimizer x^* at which the sufficient conditions hold.

We begin with the standard taxonomy of convergence rates [35], [74] and [89]:

Definition. *Let $\{x_n\} \subset \mathbb{R}^n$ and $x^* \in \mathbb{R}^n$. Then,*

(i) $x_n \to x^*$ q-quadratically if $x_n \to x^*$ and there is $K > 0$ such that

$$\|x_{n+1} - x^*\| \leq K\|x_n - x^*\|^2.$$

(ii) $x_n \to x^*$ q-superlinearly with q-order $\alpha > 1$ if $x_n \to x^*$ and there is $K > 0$ such that

$$\|x_{n+1} - x^*\| \leq K\|x_n - x^*\|^\alpha.$$

(iii) $x_n \to x^*$ q-superlinearly if

$$\lim_{n\to\infty} \frac{\|x_{n+1} - x^*\|}{\|x_n - x^*\|} = 0.$$

(iv) $x_n \to x^*$ q-linearly with q-factor $\sigma \in (0, 1)$ if

$$\|x_{n+1} - x^*\| \leq \sigma\|x_n - x^*\|$$

for n sufficiently large.

Definition. *An iterative method for computing x^* is said to be locally (q-quadratically, q-superlinearly, q-linearly) convergent if the iterates converge to x^* (q-quadratically, q-superlinearly, q-linearly) given that the initial data for the iteration is sufficiently good.*

Note that a q-superlinearly convergent sequence is also q-linearly convergent with q-factor σ for any $\sigma > 0$. A q-quadratically convergent sequence is q-superlinearly convergent with q-order of 2.

In some cases, the accuracy of the iteration can be improved by means that are external to the algorithm, say, by evaluation of the objective function and its gradient with increasing accuracy as the iteration progresses. In such cases, one has no guarantee that the accuracy of the iteration is monotonically increasing, but only that the accuracy of the results is improving at a rate determined by the improving accuracy in the function-gradient evaluations. The concept of r-type convergence captures this effect.

Definition. *Let $\{x_n\} \subset \mathbb{R}^n$ and $x^* \in \mathbb{R}^n$. Then, $\{x_n\}$ converges to x^* r-(quadratically, superlinearly, linearly) if there is a sequence $\{\xi_n\} \subset \mathbb{R}$ converging q-(quadratically, superlinearly, linearly) to 0 such that*

$$\|x_n - x^*\| \leq \xi_n.$$

Moreover, we say that $\{x_n\}$ converges r-superlinearly with r-order $\alpha > 1$ if $\xi_n \to 0$ q-superlinearly with q-order α. —

2.8 Minimization of Functionals

The function $f(x)$ such that in the following minimization problem:

$$\min_{x \in \mathbb{R}^n} f(x), \tag{2.19}$$

is a particular example of a functional, that is, a transformation from some linear space X into the space of real scalars. In (2.19), the linear space X is $\mathbb{R}^n$

with distance, inner-product, and the like, suitably defined. The familiar concepts of differentiation, distance and inner-product can be defined on most linear spaces. Consequently, nearly all the algorithms described in this thesis can be extended, at least theoretically, to the minimization of more general functionals, for example, to the following problem [86]:

$$\min_{x(t)\in D} \left\{ V(x(t)) = \int_0^1 f(x(t), t)dt \right\}, \tag{2.20}$$

where $D = \{x(t) : x(t) \in W'; x(0) = a, x(1) = b\}$, W' is the linear space of real valued absolutely continuous functions defined on $[0, 1]$, and $f(x(t), t)$ is some prescribed non-linear function of $x(t)$ and t.

Although the algorithms for minimizing the type of functional appearing in (2.20) are not discussed in this thesis, it is important to realize that the relative merits of a particular implementation of an algorithm on a computer will change when the type of functional changes. A good example of this is provided by the conjugate gradient methods (Chapter 5) which are much easier to adapt to solve (2.20) than the quasi-Newton methods (Chapter 6).

Chapter 3

Steepest Descent Method

3.1 Background

We begin the study of optimization problems with the classical problem of unconstrained minimization of a smooth function:

$$\min f(x), \quad x \in \mathbb{R}^n.$$

We focus our attention on this problem not only because of its importance, but also because, due to its simplicity, it clearly exhibits the main features of the nature of optimization problems and theoretical foundations thereof.

Let us consider the steepest descent method [21] first. The steepest descent method just uses the negative direction of its gradient to be the direction in the minimization problem. This method is not recommended for general use, but is included because of its simplicity and also because it helps to give some insight into the more sophisticated methods. In common with all gradient methods, the steepest descent method (ascent for maximizing problems) is iterative, proceed-

ing from an initial approximation x_0 for the minimizing point to successive points $x_1, x_2, ...$, until some stopping condition is satisfied.

Let us see the idea of this method. Let $f(x)$ be the differentiable function in the neighborhood of x_k, and $g_k = \nabla f(x_k) \neq 0$. By the Taylor expression:

$$f(x) = f(x_k) + (x - x_k)^T \nabla f(x_k) + o(||x - x_k||) \tag{3.1}$$

So, we know that if we denote $x - x_k = \alpha d_k$, then d_k is the descent direction which satisfies $d_k^T g_k < 0$. When α is fixed, if the value of $d_k^T g_k$ is small, that means the value $-d_k^T g_k$ is large, so the function decreases more rapidly. From the Cauchy-Schwartz inequality,

$$|d_k^T g_k| \leq ||d_k|| ||g_k|| \tag{3.2}$$

Hence, when $d_k = -g_k$, $d_k^T g_k$ is minimum [9]. So, $-g_k$ is called the descent direction.

The iterative method is as follows:

$$x_{k+1} = x_k - \alpha_k g_k.$$

Algorithm 3.1 (Steepest Descent Method)

Step 1: Given $x_0 \in \mathbb{R}^n$, $0 < \varepsilon \ll 1$, $k := 0$.

Step 2: Calculate $d_k = -g_k$, *if* $||g_k|| \leq \varepsilon$, *then stop.*

Step 3: Find the steplength α_k *by the line search*

$$f(x_k + \alpha_k d_k) = \min_{\alpha > 0} f(x_k + \alpha d_k);$$

Step 4: Calculate $x_{k+1} = x_k + \alpha_k d_k$.

Step 5: Set $k := k + 1$, *then go back to Step 2.*

3.2 Line Search Method and the Armijo Rule

We introduce some new concepts [75] so that our proof of convergence of Algorithm 3.1 will also apply to a significantly more general class of algorithms.

Definition. *A vector $d \in \mathbb{R}^n$ is a descent direction for f at x if*

$$\left.\frac{df(x+td)}{dt}\right|_{t=0} = \nabla f(x)^T d < 0.$$

Clearly, the steepest descent direction $d = -\nabla f(x)$ is a descent direction. A line search algorithm searches for decrease in f in a descent direction, using the Armijo rule for steplength control, unless $\nabla f(x) = 0$.

We will consider descent directions based on quadratic models of f of the form

$$m(x) := f(x_k) + \nabla f(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T H_k (x - x_k),$$

where H_k, which is sometimes called the model Hessian, is symmetric and positive definite. Let $d = x - x_k$ be such that $m(x)$ is minimized. Hence,

$$\nabla m(x) = \nabla f(x_k) + H_k(x - x_k) = 0$$

and hence

$$d = -H_k^{-1} \nabla f(x_k) \tag{3.3}$$

The steepest descent direction satisfies (3.3) with $H_k = I$. However, for some other methods, the direction may fail to be a descent direction if x is far from a minimizer since $H_k = \nabla^2 f(x_k)$ may not be symmetric and positive definite. Therefore, unlike the case for nonlinear equations, those methods are not generally good global method, even with line search, and they must be modified to make sure that the model Hessians are symmetric and positive definite (see [54],

[57], [43] and [108]).

To make the steepest descent method succeed, it is important to choose the steplength α. Let us see the following example first [34]:

Consider the function $f(x) = x^2$ with the initial guess $x_0 = 2$. We choose $d_k = (-1)^{k+1}$, which is a descent direction of f. We also choose the steplength $\alpha_k = 2 + 3 \cdot 2^{-(k+1)}$. Then,

$$\begin{aligned}
x_k &= x_{k-1} + \alpha_{k-1} d_{k-1} \\
&= x_{k-1} + (2 + 3 \cdot 2^{-k})(-1)^k \\
&= x_{k-1} + 2(-1)^k + 3((-1)^k 2^{-k}) \\
&= \ldots \\
&= x_0 + 2[(-1)^k + \ldots + (-1)] + 3[(-1)^k 2^{-k} + \ldots + (-1)2^{-1}] \\
&= 2 + 2\left[\frac{(-1)(1-(-1)^k)}{1-(-1)}\right] + 3\left[\frac{-\frac{1}{2}(1-(-\frac{1}{2})^k)}{1-(-\frac{1}{2})}\right] \\
&= 2 + [-1 + (-1)^k] + [-1 + (-\frac{1}{2})^k] \\
&= (-1)^k(1 + 2^{-k})
\end{aligned}$$

Hence, we have $f(x_k) = (1 + 2^{-k})^2$.

Here are some results of the first few iterations:

k	0	1	2	3	...
x_k	2	$-\frac{3}{2}$	$\frac{5}{4}$	$-\frac{9}{8}$	...
$f(x_k)$	4	$\frac{9}{4}$	$\frac{25}{16}$	$\frac{81}{64}$	...
d_k	-1	1	-1	1	...

Note that the minimizer of f is $(0, 0)$. Although the sequence $\{f(x_k)\}$ is

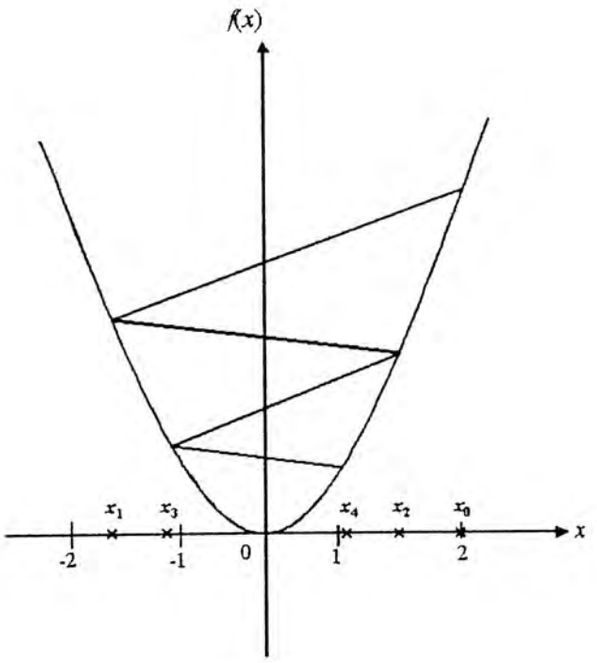

Figure 3.1: The sequence $f(x_n) = x_n^2$ is strictly decreasing, but does not converge to the minimizer $(0, 0)$.

strictly decreasing, we observe that:

$$\lim_{n\to\infty} f(x_n) = 1 \quad \text{and}$$

$$\lim_{n\to\infty} x_n = \begin{cases} 1, & \text{if } n \text{ is even;} \\ -1, & \text{if } n \text{ is odd.} \end{cases}$$

Therefore, this scheme does not converge to the minimizer since the decreasing in the values of $f(x_n)$ is very small comparing with the steplength α_k.

Now, we should find some rules for the steplength α_k to overcome this problem. Let us see the iterative formula first:

$$\begin{aligned} x_{k+1} &= x_k - \alpha_k \nabla f(x_k) \\ f(x_{k+1}) &= f(x_k - \alpha_k \nabla f(x_k)) \end{aligned}$$

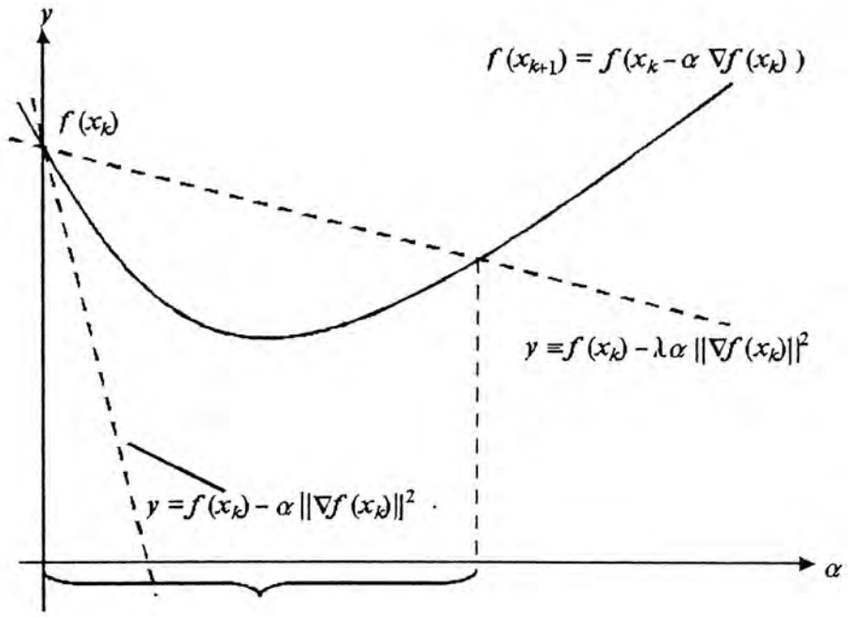

Figure 3.2: Permissible values of α under Armijo rule

By Taylor's expression, we have:

$$\begin{aligned} f(x_{k+1}) &= f(x_k) + \nabla f(x_k) \cdot (-\alpha_k \nabla f(x_k)) + o(\|\nabla f(x_k)\|) \\ &= f(x_k) - \alpha_k \|\nabla f(x_k)\|^2 + o(\|\nabla f(x_k)\|) \\ f(x_{k+1}) - f(x_k) &= -\alpha_k \|\nabla f(x_k)\|^2 + o(\|\nabla f(x_k)\|) \\ &\leq -\lambda\alpha_k \|\nabla f(x_k)\|^2 \quad \text{, for some } \lambda \in (0,1). \end{aligned}$$

Hence, we can fix this by requiring that the average rate of decreasing from $f(x_k)$ to $f(x_{k+1})$ be at least some prescribed fraction of the initial rate of decreasing.

That is, take $\lambda \in (0,1)$, choose α_k from among those $\alpha > 0$ which satisfies:

$$f(x_{k+1}) - f(x_k) \leq -\lambda\alpha_k \|\nabla f(x_k)\|^2 \tag{3.4}$$

The above rule is the so-called *Armijo Rule* ([2], [61]).

In Figure 3.2, It is clearly to see that the range of the permissible values of α under Armijo rule is much larger than the original ones.

3.3 Steplength Control with Polynomial Models

The steplength reduction scheme for α in Algorithm 3.1 is too crude. If α is too large, too many steplength reductions may be needed before a step is accepted. If α is too small, the progress of the entire iteration may be retarded. Hence, we will now address this problem in the following way: we will construct polynomial models of f along the descent direction to predict an optimal factor by which to reduce the step ([30], [35], [56], [87], [57]).

3.3.1 Quadratic Polynomial Model

Having computed a descent direction d from x_0, one must decide on a steplength reduction scheme for iterations in which (3.4) fails for $\alpha = 1$. A common approach, which is called *steplength control with polynomial models* (or called *backtracking line-search framework*), is to model

$$\xi(\alpha) = f(x_0 - \alpha \nabla f(x_0))$$

by a quadratic polynomial. The data on hand initially are:

(i) $\xi(0) = f(x_0)$

(ii) $\xi'(0) = \nabla f(x_0 - \alpha \nabla f(x_0)) \cdot (-\nabla f(x_0))|_{\alpha=0} = -\|\nabla f(x_0)\|^2$

(iii) $\xi(1) = f(x_0 - \nabla f(x_0))$

From the above data, we have the following two facts:

1. $\xi'(0) = -\|\nabla f(x_0)\|^2 < 0$.

2. Since $\alpha = 1$ does not satisfies (3.4), then

$$\begin{aligned} \xi(1) &= f(x_0 - \nabla f(x_0)) \\ &= f(x_1) \\ &\geq f(x_0) - \lambda \|\nabla f(x_0)\|^2 \\ &= \xi(0) + \lambda \xi'(0). \end{aligned}$$

Hence,

$$\xi(1) - \xi(0) \geq \lambda \xi'(0). \tag{3.5}$$

By (i) to (iii), we can approximate ξ by a quadratic polynomial model:

$$p(\alpha) = \xi(0) + \xi'(0)\alpha + (\xi(1) - \xi(0) - \xi'(0))\alpha^2.$$

Since the leading coefficient of $p(\alpha)$ is

$$\begin{aligned} \xi(1) - \xi(0) - \xi'(0) &\geq \lambda \xi'(0) - \xi'(0) \\ &= (\lambda - 1)\xi'(0) \\ &> 0 \end{aligned}$$

for $\lambda \in (0, 1)$ and $\xi'(0) < 0$.

Hence, p is a concave upward function, which has a unique minimizer, denoted by α_p, which can be computed by:

$$\alpha_p = -\frac{\xi'(0)}{2(\xi(1) - \xi(0) - \xi'(0))}. \tag{3.6}$$

We see that $\xi'(0) < 0$ and $\xi(1) - \xi(0) - \xi'(0) > 0$, then we have:

$$\alpha_p > 0.$$

Now, we can take α_p as the new value of α to compute.

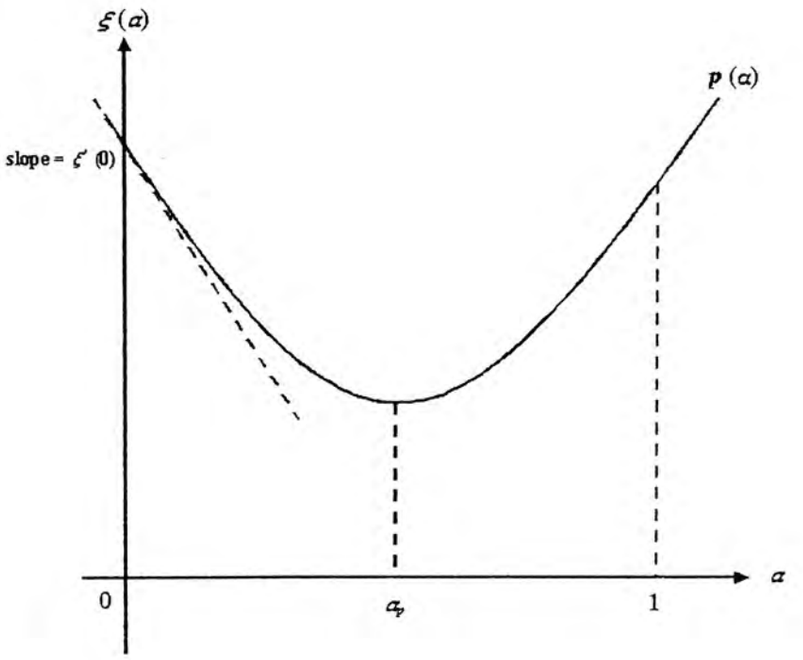

Figure 3.3: Backtracking at the first iteration, using the quadratic model

3.3.2 Safeguarding

Note that the steplength α cannot be too large or too small. Hence, we require the following safeguarding [12]:

If a steplength α_0 has been rejected, that is (3.4) fail with $\alpha = \alpha_0$, construct

$$\alpha_+ \in [\beta_{low}, \beta_{high}] \tag{3.7}$$

where $0 < \beta_{low} \leq \beta_{high} < 1$.

Hence, we have:

$$\alpha_+ = \begin{cases} \beta_{low}, & \text{if } \alpha_p \leq \beta_{low}, \\ \alpha_p, & \text{if } \beta_{low} < \alpha_p < \beta_{high}, \\ \beta_{high}, & \text{if } \alpha_p \geq \beta_{high}. \end{cases} \tag{3.8}$$

There are some advantages of using the safeguarding when the quadratic model $p(\alpha)$ is poor, they are:

1. The steplength α_+ is so small, that is $\alpha_+ \approx 0$, then it makes the iteration stagnating.

2. The steplength α_+ is so large, that is $\alpha_+ \approx 1$, then it needs many steplength reductions to make Armijo rule hold.

3.3.3 Cubic Polynomial Model

Now, we have the value of α_+, we can compute $x_1 = x_0 - \alpha_+ \nabla f(x_0)$ and $f(x_1)$. But also, we have to check whether the new value of the steplength α_+ is satisfies (3.4) or not. That is,

$$f(x_1) - f(x_0) \leq -\lambda \alpha_+ \|\nabla f(x_0)\|^2$$

is valid or not.

If the first reduced value of α does not satisfy (3.4), we should have the additional reductions.

Note that we now have four pieces of information about $\xi(\alpha)$, they are:

(i) $\xi(0) = f(x_0)$

(ii) $\xi'(0) = -\|\nabla f(x_0)\|^2$

(iii) $\xi(\alpha_0) = f(x_0 - \alpha_0 \nabla f(x_0))$

(iv) $\xi(\alpha_+) = f(x_0 - \alpha_+ \nabla f(x_0))$

where $\alpha_+ < \alpha_0$ is the most recent value of α which fails to satisfy (3.4). These are sufficient data to approximate ξ with a cubic polynomial model [75]:

$$q(\alpha) = \xi(0) + \xi'(0)\alpha + c_2\alpha^2 + d_2\alpha^3.$$

To determine the values of c_2 and d_2, we use:

$$\begin{cases} q(\alpha_0) = \xi(\alpha_0) \\ \\ q(\alpha_+) = \xi(\alpha_+) \end{cases}$$

and (iii) and (iv) to derive

$$\begin{cases} c_2\alpha_0^2 + d_2\alpha_0^3 = \xi(\alpha_0) - \xi(0) - \xi'(0)\alpha_0 \\ \\ c_2\alpha_+^2 + d_2\alpha_+^3 = \xi(\alpha_+) - \xi(0) - \xi'(0)\alpha_+ \end{cases}$$

This gives the following system of linear equations:

$$\begin{pmatrix} \alpha_0^2 & \alpha_0^3 \\ \\ \alpha_+^2 & \alpha_+^3 \end{pmatrix} \begin{pmatrix} c_2 \\ \\ d_2 \end{pmatrix} = \begin{pmatrix} \xi(\alpha_0) - \xi(0) - \xi'(0)\alpha_0 \\ \\ \xi(\alpha_+) - \xi(0) - \xi'(0)\alpha_+ \end{pmatrix}$$

Note that the matrix

$$\begin{pmatrix} \alpha_0^2 & \alpha_0^3 \\ \\ \alpha_+^2 & \alpha_+^3 \end{pmatrix}$$

is non-singular since

$$\det \begin{pmatrix} \alpha_0^2 & \alpha_0^3 \\ \\ \alpha_+^2 & \alpha_+^3 \end{pmatrix} = \alpha_0^2\alpha_+^2(\alpha_+ - \alpha_0) \neq 0.$$

Hence, the cubic approximation model for ξ is:

$$q(\alpha) = \xi(0) + \xi'(0)\alpha + c_2\alpha^2 + d_2\alpha^3$$

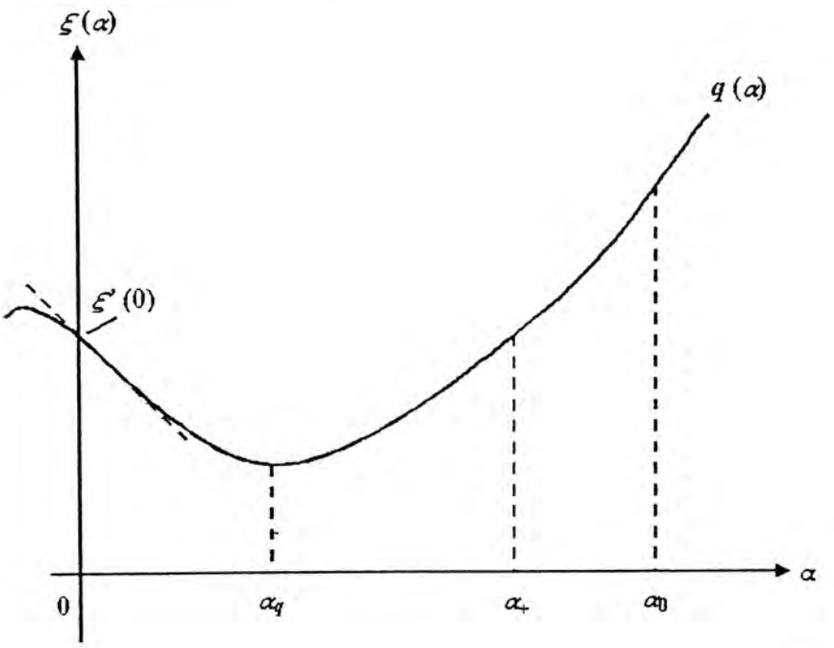

Figure 3.4: $q(\alpha)$ has a unique minimizer when $\xi'' > 0$

where

$$\begin{pmatrix} c_2 \\ d_2 \end{pmatrix} = \begin{pmatrix} \alpha_0^2 & \alpha_0^3 \\ \alpha_+^2 & \alpha_+^3 \end{pmatrix}^{-1} \begin{pmatrix} \xi(\alpha_0) - \xi(0) - \xi'(0)\alpha_0 \\ \xi(\alpha_+) - \xi(0) - \xi'(0)\alpha_+ \end{pmatrix}.$$

Here, $q(\alpha)$ must have a unique minimizer. Note that $\xi'(0) < 0$ and $\xi(\alpha_+) < \xi(\alpha_0)$. Let us discuss this in the following two cases [34]:

Case 1: When $\xi(\alpha)$ is concave upward, where $\alpha_+ < \alpha < \alpha_0$, the graph would like Figure 3.4. From Figure 3.4, it is clearly to see that $q(\alpha)$ has a unique minimizer.

Case 2: When $\xi(\alpha)$ is concave downward, where $\alpha_+ < \alpha < \alpha_0$, the graph would like Figure 3.5. From Figure 3.5, it is also clearly to see that $q(\alpha)$ has a unique minimizer.

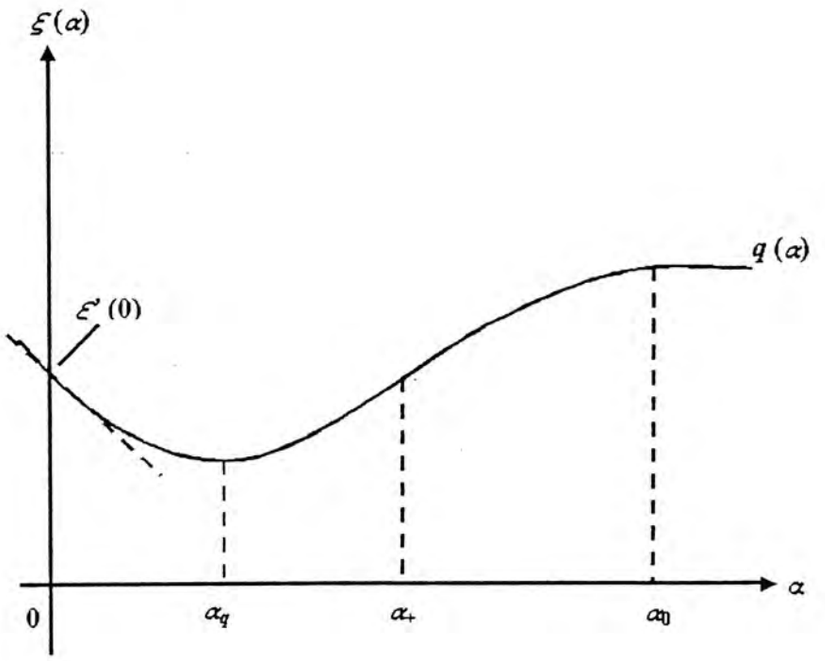

Figure 3.5: $q(\alpha)$ has a unique minimizer when $\xi'' < 0$

From the above two cases, $q(\alpha)$ has a unique minimizer, dented by α_q, which can be computed by the following formula [35]:

$$\alpha_q = \frac{-c_2 + \sqrt{c_2^2 - 3d_2\xi'(0)}}{3d_2},$$

where we have used the cubic model instead of the quadratic one, because it can more accurately model situations where f has negative curature, which are likely when Armijo rule has failed for two positive values of α.

3.3.4 General Line Search Strategy

Note that the steplength α cannot be too large or too small. Hence, we require the following general line search strategy [75]:

If a steplength α_0 has been rejected, that is (3.4) fails with $\alpha = \alpha_0$, construct

$$\alpha_+ \in [\beta_{low}\alpha_0, \beta_{high}\alpha_0] \tag{3.9}$$

where $0 < \beta_{low} \leq \beta_{high} < 1$.

The choice $\beta = \beta_{low} = \beta_{high}$ is the simple rule in Algorithm 3.1.

Hence, we have:

$$\alpha'_+ = \begin{cases} \beta_{low}\alpha_+, & \text{if } \alpha_q \leq \beta_{low}\alpha_+, \\ \alpha_q, & \text{if } \beta_{low}\alpha_+ < \alpha_q < \beta_{high}\alpha_+, \\ \beta_{high}\alpha_+, & \text{if } \alpha_q \geq \beta_{high}\alpha_+. \end{cases} \tag{3.10}$$

Here, we use this general line search strategy instead of the exact line search, in which α is the exact minimum of $f(x_0+\alpha d)$, since the exact line search method has the following disadvantages:

1. It is more expensive.
2. It degrades the performance of the whole algorithm.

Algorithm 3.2 (General Line Search Strategy)

Step 1: *Given* $x_0 \in \mathbb{R}^n$, $k := 0$ *and* $k_{max} > 0$.

Step 2: *Calculate* $f(x_k)$ *and* $\nabla f(x_k)$.

Step 3: *Construct a symmetric and positive definite matrix* H *and solve (3.3) to obtain a descent direction* d.

Step 4: *Beginning with* $\alpha = 1$, *repeatedly reduce* α *using any strategy that satisfies (3.9) until (3.4) holds.*

Step 5: *Calculate* $x_{k+1} = x_k + \alpha_k d_k$.

Step 6: Set $k := k + 1$.

Step 7: If $k < k_{max}$, *then go back to Step 2; otherwise, stop.*

Now, we have the value of α'_+, we can compute $x_1 = x_0 - \alpha'_+ \nabla f(x_0)$ and $f(x_1)$. But again, we have to check if the new value of the steplength α_+ satisfies (3.4) or not. That is,

$$f(x_1) - f(x_0) \leq -\lambda \alpha'_+ \|\nabla f(x_0)\|^2$$

is valid or not.

If α'_+ does not satisfy the Armijo rule, and now we have the most two recent values of α, that is α_+ and α'_+, then we can repeat the whole process of the cubic polynomial model, to find a new value of α. Finally, repeat this process again and again until the value of α satisfies (3.4).

3.3.5 Algorithm of Steepest Descent Method

In this part, we will conclude the whole process of the steepest descent method which has been discussed in the previous subsections. We summarize in the following algorithm ([113], [119]):

Algorithm 3.3 (Steepest Descent Method)

Step 0: Given the function $f : D \subset \mathbb{R}^n \to \mathbb{R}$ *and the initial guess* $x_0 \in D$.

Step 1: The 1st iteration

(i) Compute $f(x_0)$ *and* $\nabla f(x_0)$.

(ii) Set $\alpha_0 = 1$.

(iii) Compute $x_1 = x_0 - \alpha_0 \nabla f(x_0)$ *and* $f(x_1)$.

(iv) If α_0 does not satisfy the Armijo rule, that is,

$$f(x_1) - f(x_0) > -\lambda\alpha_0\|\nabla f(x_0)\|^2$$

for some $\lambda \in (0,1)$, then use the following backtracking line-search framework:

(1) Let $\xi(\alpha) = f(x_0 - \alpha\nabla f(x_0))$.

(2) Compute

$$\alpha_p = -\frac{\xi'(0)}{2(\xi(1) - \xi(0) - \xi'(0))}.$$

(3) Safeguarding

$$\alpha_+ = \begin{cases} \beta_{low}, & \text{if } \alpha_p \leq \beta_{low}, \\ \alpha_p, & \text{if } \beta_{low} < \alpha_p < \beta_{high}, \\ \beta_{high}, & \text{if } \alpha_p \geq \beta_{high}. \end{cases}$$

for some $0 < \beta_{low} \leq \beta_{high} < 1$.

(4) Compute $x_1 = x_0 - \alpha_+\nabla f(x_0)$ and $f(x_1)$.

(v) If α_+ still does not satisfy the Armijo rule, that is,

$$f(x_1) - f(x_0) > -\lambda\alpha_+\|f(x_0)\|^2,$$

then

(1) Compute

$$\alpha_q = \frac{-c_2 + \sqrt{c_2^2 - 3d_2\xi'(0)}}{3d_2},$$

where

$$\begin{pmatrix} c_2 \\ d_2 \end{pmatrix} = \begin{pmatrix} \alpha_0^2 & \alpha_0^3 \\ \alpha_+^2 & \alpha_+^3 \end{pmatrix}^{-1} \begin{pmatrix} \xi(\alpha_0) - \xi(0) - \xi'(0)\alpha_0 \\ \xi(\alpha_+) - \xi(0) - \xi'(0)\alpha_+ \end{pmatrix}.$$

(2) General line search strategy

$$\alpha'_+ = \begin{cases} \beta_{low}\alpha_+, & \text{if } \alpha_q \leq \beta_{low}\alpha_+, \\ \alpha_q, & \text{if } \beta_{low}\alpha_+ < \alpha_q < \beta_{high}\alpha_+, \\ \beta_{high}\alpha_+, & \text{if } \alpha_q \geq \beta_{high}\alpha_+. \end{cases}$$

for some $0 < \beta_{low} \leq \beta_{high} < 1$.

(3) Compute $x_1 = x_0 - \alpha'_+ \nabla f(x_0)$ *and* $f(x_1)$.

(vi) When α'_+ *still does not satisfy the Armijo rule, then we use the most two recent values of* α, *and repeat the process in (v) with* α_0 *and* α_+ *repeated by these 2 most recent values of* α.

(vii) Repeat (v) until the value of α *satisfies Armijo rule.*

Step 2: The 2nd iteration

(i) Compute $f(x_1)$ *and* $\nabla f(x_1)$.

(ii) Set $\alpha_0 = 1$.

(iii) Compute $x_2 = x_1 - \alpha_0 \nabla f(x_1)$ *and* $f(x_2)$.

(iv) Do the backtracking line-search framework in Step 1 (iv), (v), (vi) and (vii) again by considering the following function:

$$\xi(\alpha) = f(x_1 - \alpha \nabla f(x_1)).$$

(v) Finally, $x_2 = x_1 - \alpha_+ \nabla f(x_1)$, *where* α_+ *is the most recent value of* α *which satisfies the Armijo rule.*

In general, for $k \geq 1$,

Step k: The kth iteration

(i) Compute $f(x_{k-1})$ *and* $\nabla f(x_{k-1})$.

(ii) *Set* $\alpha_0 = 1$.

(iii) *Compute* $x_k = x_{k-1} - \alpha_0 \nabla f(x_{k-1})$ *and* $f(x_k)$.

(iv) *Do the backtracking line-search framework in Step 1 (iv), (v), (vi) and (vii) again by considering the following function:*

$$\xi(\alpha) = f(x_{k-1} - \alpha \nabla f(x_{k-1})).$$

(v) *Finally,* $x_k = x_{k-1} - \alpha_+ \nabla f(x_{k-1})$, *where* α_+ *is the most recent value of* α *which satisfies the Armijo rule.*

3.4 Advantages of the Armijo Rule

There are many sufficient decrease conditions. But we use the Armijo rule since this decrease condition can be satisfied in *finite* many steps.

Let us see the following lemmas and theorem [75]:

Lemma 3.1 *Let* H *be symmetric and positive definite with smallest and largest eigenvalues* $0 < \lambda_n < \lambda_1$. *Then, for all* $z \in \mathbb{R}^n$,

$$\lambda_1^{-1}\|z\|^2 \leq z^T H^{-1} z \leq \lambda_n^{-1}\|z\|^2.$$

Lemma 3.2 *Assume that* ∇f *is Lipschitz continuous with Lipschitz constant* L. *Let* $\lambda \in (0,1)$, $x \in \mathbb{R}^n$, *and* H *be an symmetric and positive definite matrix. Let* $\lambda_n > 0$ *be the smallest and* $\lambda_1 \geq \lambda_n$ *the largest eigenvalues of* H. *Let* d *be given by (3.3). Assume that* $\nabla f(x) \neq 0$. *Then, (3.4) holds for any* α *such that*

$$0 < \alpha \leq \frac{2\lambda_n(1-\lambda)}{L\kappa(H)}. \tag{3.11}$$

Lemma 3.3 *Let ∇f be Lipschitz continuous with Lipschitz constant L. Let $\{x_k\}$ be the iteration given by Algorithm 3.2 with symmetric and positive definite matrices H_k that satisfy*

$$\kappa(H_k) \leq \bar{\kappa} \tag{3.12}$$

for all k. Then, the steps

$$s_k = x_{k+1} - x_k = \lambda_k d_k = -\lambda_k H_k^{-1} \nabla f$$

satisfy

$$\lambda_k \geq \bar{\lambda} = \frac{2\beta_{low}\lambda_n(1-\lambda)}{L\bar{\kappa}} \tag{3.13}$$

and at most

$$m = \log\left(\frac{2\lambda_n(1-\lambda)}{L\bar{\kappa}}\right) / \log(\beta_{high}) \tag{3.14}$$

steplength reductions will be required.

The convergence theorem for Algorithm 3.2 says that if the condition numbers of the matrices H and the norms of the iterates remain bounded, then every limit point of the iteration is a stationary point. Boundedness of the sequence of iterates implies that there will be limit points, but there is no guarantee that there is a unique limit point.

Theorem 3.4 *Let ∇f be Lipschitz continuous with Lipschitz constant L. Assume that the matrices H_k are symmetric and positive definite as well as that there are $\bar{\kappa}$ and α_l such that $\kappa(H_k) \leq \bar{\kappa}$, and $\|H_k\| \leq \alpha_l$ for all k. Then, either $f(x_k)$ is unbounded from below or*

$$\lim_{k\to\infty} \nabla f(x_k) = 0 \tag{3.15}$$

and hence any limit point of the sequence of iterates produced by Armijo rule in the Steepest Descent Method is a stationary point.
In particular, if $f(x_k)$ is bounded from below and $x_{k_l} \to x^*$ is any convergent subsequence of $\{x_k\}$, then $\nabla f(x^*) = 0$.

Proof: By construction, $f(x_k)$ is a decreasing sequence. Therefore, if $f(x_k)$ is bounded from below, then $\lim_{k\to\infty} f(x_k) = f^*$ exists and

$$\lim_{k\to\infty} f(x_{k+1}) - f(x_k) = 0 \tag{3.16}$$

By (3.4) and Lemma 3.3, we have:

$$\begin{aligned} f(x_{k+1}) - f(x_k) &< -\lambda\alpha_k \nabla f(x_k)^T H_k^{-1} \nabla f(x_k) \\ &\leq -\lambda\bar{\alpha}\alpha_l^{-1} \|\nabla f(x_k)\|^2 \\ &\leq 0. \end{aligned}$$

Hence, by (3.16),

$$\|\nabla f(x_k)\| \leq \frac{\alpha_l(f(x_k) - f(x_{k+1}))}{\lambda\bar{\alpha}} \to 0$$

as $k \to \infty$. This completes the proof. □

The analysis of the Armijo rule is valid for other line search methods [35], [59], [125] and [126]. The key points are that the sufficient decrease condition can be satisfied in finitely many steps and that the steplengths are bounded away from zero.

3.5 Convergence Analysis

Unfortunately, the methods based on the steepest descent do not enjoy good local convergence properties, even for very simple functions. This section discusses the

convergence, closely following Yuan-Sun [128]. Let us see the following lemmas first:

Lemma 3.5 *Suppose $\alpha_k > 0$ is the solution of*

$$f(x_{k+1}) = f(x_k + \alpha_k d_k) = \min_{\alpha>0} f(x_k + \alpha d_k).$$

If for all $\alpha > 0$, there exists $M > 0$ such that

$$\|\nabla^2 f(x_k + \alpha_k d_k)\| \leq M,$$

then we have:

$$f(x_k) - f(x_k + \alpha_k d_k) \geq \frac{1}{2M}\|\nabla f(x_k)\|^2 \cos^2\langle d_k, -\nabla f(x_k)\rangle,$$

where

$$\cos\langle d_k, -\nabla f(x_k)\rangle = -\frac{d_k^T \nabla f(x_k)}{\|d_k\|\|\nabla f(x_k)\|}.$$

Lemma 3.6 *Let $\varphi(x)$ be a function which is twice differentiable on the closed interval $[0, b]$ such that $\varphi'(0) < 0$. If $x^* \in (0, b)$ is the minimizer of $\varphi(x)$ in $[0, b]$, then we have:*

$$x^* \geq \tilde{x} = -\frac{\varphi'(0)}{M} \tag{3.17}$$

where $\varphi''(x) \leq M$, for all $x \in [0, b]$.

Lemma 3.7 *Suppose $f : D \subset \mathbb{R}^n \to \mathbb{R}$ is twice continuous differentiable. Then, for all $x, d \in \mathbb{R}^n$ and for all $\alpha \in \mathbb{R}$, we have:*

$$f(x + \alpha d) = f(x) + \alpha \nabla f(x)^T d + \alpha^2 \int_0^1 (1 - t)\left[d^T \nabla^2 f(x + \alpha dt) d\right] dt \tag{3.18}$$

Lemma 3.8 *Suppose $f(x)$ is twice continuous differentiable at the minimizer x^*. Also, for all $y \in \mathbb{R}^n$, there exists $\varepsilon > 0$ and $M > m > 0$ such that*

$$m\|y\|^2 \leq y^T \nabla^2 f(x) y \leq M\|y\|^2 \tag{3.19}$$

whenever $\|x - x^*\| < \varepsilon$. Then, we have:

(i)

$$\frac{1}{2}m\|x - x^*\|^2 \leq f(x) - f(x^*) \leq \frac{1}{2}M\|x - x^*\|^2 \tag{3.20}$$

(ii)

$$\|\nabla f(x)\| \geq \bar{m}\|x - x^*\| \tag{3.21}$$

From the above lemmas, then we have the following theorem about the rate of convergence of the line search for the steplength α_k [128]:

Theorem 3.9 *Let (x_k) be the sequence generated by the line search for the steplength α_k, and $x_k \to x^*$ as $k \to \infty$. Suppose $f(x)$ is twice continuous differentiable at x^*. Also, for all $y \in \mathbb{R}^n$, there exists $\varepsilon > 0$ and $M > m > 0$ such that*

$$m\|y\|^2 \leq y^T \nabla^2 f(x) y \leq M\|y\|^2 \tag{3.22}$$

whenever $\|x - x^*\| < \varepsilon$. Then, (x_k) converges to x^* r-linearly.

Proof: Let $\lim_{k\to\infty} x_k = x^*$, and assume that

$$\|x_k - x^*\| \leq \varepsilon, \tag{3.23}$$

for $k = 0, 1, \ldots$. Since $\|x_{k+1} - x^*\| < \varepsilon$, then there exists $\delta > 0$ such that

$$\|x_k + (\alpha_k + \delta)d_k - x^*\| = \|x_{k+1} - x^* + \delta d_k\| < \varepsilon.$$

Also, since $\varphi'(0) < 0$ and $\varphi''(0) = d_k^T G(x_k + \alpha d_k) d_k \leq M\|d_k\|^2$, then from Lemma 3.6, the minimizer α_k of the function $\varphi(\alpha) = f(x_k + \alpha d_k)$ in the interval $[0, \alpha_k + \delta]$ satisfies

$$\alpha_k \geq \bar{\alpha}_k = \frac{-\varphi'(0)}{M\|d_k\|^2} \geq \frac{\rho\|g_k\|}{M\|d_k\|}. \tag{3.24}$$

Denote

$$\bar{\alpha}_k = \frac{\rho\|g_k\|}{M\|d_k\|}, \tag{3.25}$$

such that $\bar{x}_k = x_k + \bar{\alpha}_k d_k$. Clearly, $\|\bar{x}_k - x^*\| < \varepsilon$. Using Lemma 3.7, we have:

$$\begin{aligned}
f(x_k + \alpha_k d_k) - f(x_k) &\leq f(x_k + \bar{\alpha}_k d_k) - f(x_k) \\
&= \bar{\alpha}_k g_k^T d_k + \bar{\alpha}_k{}^2 \int_0^1 (1-t) d_k^T G(x_k + \bar{\alpha}_k d_k) d_k dt \\
&\leq \bar{\alpha}_k(-\rho)\|g_k\|\|d_k\| + \frac{1}{2} M \bar{\alpha}_2{}^2 \|d_k\|^2 \\
&\leq -\frac{\rho^2}{2M}\|g_k\|^2 \\
&\leq -\frac{\rho^2}{2M} m^2 \|x_k - x^*\|^2 \\
&\leq -\left(\frac{\rho m}{M}\right)^2 [f(x_k) - f(x^*)],
\end{aligned}$$

So, we have:

$$f(x_{k+1}) - f(x^*) \leq \left[1 - \left(\frac{\rho m}{M}\right)^2\right] [f(x_k) - f(x^*)]. \tag{3.26}$$

Denote

$$\theta = \left[1 - \left(\frac{\rho m}{M}\right)^2\right]^{\frac{1}{2}}. \tag{3.27}$$

It is clear that $\theta \in (0, 1)$. So, we have:

$$\begin{aligned}
f(x_k) - f(x^*) &\leq \theta^2 [f(x_{k-1}) - f(x^*)] \\
&\leq \ldots \\
&\leq \theta^{2k} [f(x_0) - f(x^*)].
\end{aligned}$$

By Lemma 3.8, we have:

$$\begin{aligned} \|x_k - x^*\|^2 &\leq \frac{2}{m}[f(x_k) - f(x^*)] \\ &\leq \frac{2}{m}[f(x_0) - f(x^*)]\theta^{2k}. \end{aligned} \tag{3.28}$$

From (3.28), we can write

$$\|x_k - x^*\| \leq K\theta^k, \tag{3.29}$$

where $\theta < 1$ and

$$K = \sqrt{\frac{2}{m}}[f(x_0) - f(x^*)]^{\frac{1}{2}}.$$

In the statement (3.29), then the sequence (x_k) converges to x^* r-linearly. □

Now, we can have the following theorems concerning the rate of convergence of the steepest descent method [128]:

Theorem 3.10 *Suppose the function $f(x)$ satisfies all assumptions in Theorem 3.9. If the sequence (x_k) generated by the steepest descent method converges to x^*, then (x_k) converges to x^* at least r-linearly.*

Proof: Using Theorem 3.9, then we have this result. □

When we discuss the general case of the functions $f(x)$, the convergence rate of the steepest descent method 3.3 can be also describe as the following theorem:

Theorem 3.11 *Let (x_k) be the sequence generated by Steepest Descent Method and $x_k \to x^*$ as $k \to \infty$. Assume $f(x)$ is twice differentiable at the neighborhood of x^*, $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite. Then, we have*

$$\frac{f(x_{k+1}) - f(x^*)}{f(x_k) - f(x^*)} = \beta_k < 1 \tag{3.30}$$

and

$$\limsup_{k\to\infty} \beta_k \leq \frac{M-m}{M} \leq 1, \tag{3.31}$$

where M and m satisfy

$$0 < m \leq \lambda_n \leq \lambda_1 \leq M, \tag{3.32}$$

$\lambda_n = \lambda_n(\nabla^2 f(x))$ and $\lambda_1 = \lambda_1(\nabla^2 f(x))$ are the smallest and the largest eigenvalues of $\nabla^2 f(x)$ respectively.

Proof: From Lemma 3.5, then

$$\begin{aligned} [f(x_k) - f(x^*)] - [f(x_{k+1}) - f(x^*)] &= f(x_k) - f(x_{k+1}) \\ &\geq \frac{1}{2M}\|\nabla f(x_k)\|^2. \end{aligned}$$

From (3.30), we have:

$$\begin{aligned} (1-\beta_k)[f(x_k) - f(x^*)] &\geq \frac{1}{2M}\|\nabla f(x_k)\|^2 \\ \beta_k &\leq 1 - \frac{\|\nabla f(x_k)\|^2}{2M[f(x_k) - f(x^*)]} < 1. \end{aligned} \tag{3.33}$$

By assumption, when x satisfies $\|x - x^*\| < \delta$, the statement (3.32) holds, for some scalar M, m and $\delta > 0$.

Now, assume that

$$\frac{x_k - x^*}{\|x_k - x^*\|} \to \bar{d},$$

we have:

$$\begin{aligned} \|\nabla f(x_k)\|^2 &= \|x_k - x^*\|^2 \left(\|\nabla^2 f(x^*)\bar{d}\|^2 + o(1)\right) \\ f(x_k) - f(x^*) &= \frac{1}{2}\|x_k - x^*\|^2 \left(\bar{d}^T \nabla^2 f(x^*)\bar{d} + o(1)\right). \end{aligned}$$

Using the above two statements and (3.32), we have:

$$\begin{aligned} \lim_{k\to\infty} \frac{\|\nabla f(x_k)\|^2}{f(x_k) - f(x^*)} &= \frac{2\|\nabla^2 f(x^*)\bar{d}\|^2}{\bar{d}^T \nabla^2 f(x^*)\bar{d}} \\ &\geq 2m. \end{aligned} \tag{3.34}$$

Hence, from (3.33) and (3.34), we have:

$$\begin{aligned} \limsup_{k\to\infty} \beta_k &\leq 1 - \lim_{k\to\infty} \inf \frac{\|\nabla f(x_k)\|^2}{2M[f(x_k) - f(x^*)]} \\ &\leq 1 - \frac{m}{M} \\ &< 1. \end{aligned}$$

This completes the proof. □

Chapter 4

Iterative Methods Using Second Derivatives

4.1 Background

If it is given that $f(x) \in C^2$ and that its Hessian matrix H can be computed for any x, then there are some algorithms which can utilize this knowledge. These algorithms are nearly all variants of Newton's method. In this chapter, some better-known algorithms using the second derivative (or Hessian matrix H) are discussed.

Usually, refer to Murray [86], even if H is available, we could use one of the many algorithms for which this knowledge is not required. The reason for doing this is to alleviate the need to do the analytical differentiation and subsequent computer programming. The disadvantage can be partly answered by the availability of computer programs that will perform analytical differentiation. Although these programs may not produce an efficient code they are more reliable

than programs based on differentiation by hand. The need for a computer program to evaluate H is an additional feature which does not arise in algorithms which do not require H. To justify the use of a second derivative method, there must therefore be some advantages. One that can be appreciated immediately is that it is nearly always possible to check whether x^* satisfies both the necessary and sufficient conditions for a strong local minimum. The main hope is that second derivative methods will prove more reliable and take significantly fewer iterations than alternative methods.

4.2 Newton's Method

4.2.1 Basic Concepts

In the steepest descent method, the notion of local linear approximation of the objective function $f(x)$ is basic. If the function is twice differentiable, one may naturally try to use its quadratic approximation at a point x_k, that is the function

$$\tilde{f}(x) = f(x_k) + (x - x_k)^T \nabla f(x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k). \qquad (4.1)$$

In the steepest descent method, the next iteration x_{k+1} is sought under the condition that the linear approximation be a minimum point under the additional constraints of being near to x_k. For a quadratic approximation one can try to impose no restrictions of this kind, since for $\nabla^2 f(x_k) > 0$, the function $\tilde{f}(x)$ attains an unconstrained minimum.

Let us take a minimum point of $\tilde{f}(x)$ as the new approximation [94]:

$$\min_{x \in \mathbb{R}^n} \tilde{f}(x)$$

Then, we have:

$$x_{k+1} = x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k). \tag{4.2}$$

Another approach [94] to arrive at this method is that the minimum point must be a solution of the system of n equations with n variables:

$$\nabla f(x) = 0. \tag{4.3}$$

One of the basic methods for solving this system is Newton's method, which consists in linearizing the equations at a point x_k and solving the linearized system. This linearized system in the given case has the form

$$\nabla f(x_k) + \nabla^2 f(x_k) \cdot (x - x_k) = 0, \tag{4.4}$$

whose solution x_{k+1} is again given by (4.2).

Algorithm 4.1 (Newton's Method)

Step 1: Given $x_0 \in \mathbb{R}^n$ *and* $k := 0$.

Step 2: Calculate $\nabla f(x_k)$ *and* $\nabla^2 f(x_k)$.

Step 3: Calculate $x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$.

Step 4: Set $k := k + 1$, *then go back to Step 2.*

4.2.2 Convergence Analysis of Newton's Method

With refer to [89], we have the following convergence theorems of Newton's method:

Theorem 4.1 *Assume that* $f(x)$ *is twice differentiable,* $\nabla^2 f(x)$ *satisfies a Lipschitz condition with constant* L, *and* $f(x)$ *is strongly convex with constant* l, *and*

the following condition holds:

$$q = \frac{L}{2l^2}\|\nabla f(x_0)\| < 1. \tag{4.5}$$

Then, the method (4.2) converges to the global minimum point x^* with the quadratic rate:

$$\|x_k - x^*\| \leq \frac{2l}{L}q^{2^k}. \tag{4.6}$$

Proof: Since $\nabla^2 f(x)$ satisfies the Lipschitz condition with constant L on $[x, x+y]$, that is:

$$\begin{aligned} & \|\nabla^2 f(u) - \nabla^2 f(v)\| \leq L\|u - v\|, \quad u, v \in [x, x+y], \\ \text{then} \quad & \|\nabla f(x+y) - \nabla f(x) - \nabla^2 f(x)y\| \leq \frac{L}{2}\|y\|^2 \end{aligned}$$

where

$$x = x_k, \quad \text{and} \quad y = -[\nabla^2 f(x_k)]^{-1}\nabla f(x_k).$$

Then, $x + y = x_{k+1}$ and

$$\begin{aligned} \|\nabla f(x_{k+1})\| &\leq \frac{L}{2}\|[\nabla^2 f(x_k)]^{-1}\nabla f(x_k)\|^2 \\ &\leq \frac{L}{2}\|\nabla^2 f(x_k)\|^2\|\nabla f(x_k)\|^2. \end{aligned}$$

Since $f(x)$ is strongly convex with constant l, it is equivalent to the condition

$$\begin{aligned} \nabla^2 f(x) &\geq lI, \quad \text{for all } x, \\ \|\nabla f(x)\| &\geq l\|x - x^*\|. \end{aligned}$$

Then,

$$[\nabla^2 f(x_k)]^{-1} \leq \frac{1}{l}I \quad \text{and} \quad \|[\nabla^2 f(x_k)]^{-1}\| \leq \frac{1}{l},$$

that is,

$$\|\nabla f(x_{k+1})\| \leq \frac{L}{2l^2}\|\nabla f(x_k)\|^2.$$

Iterating the above inequality, then we have:

$$\begin{aligned} \|\nabla f(x_k)\| &\leq \frac{2l^2}{L}\left(\frac{L}{2l^2}\|\nabla f(x_0)\|\right)^{2^k} \\ &= \frac{2l^2}{L}q^{2^k}. \end{aligned} \tag{4.7}$$

Since for $\|\nabla f(x_k)\| \geq \|x_k - x^*\|$. Applying this statement to (4.7), then we have:

$$\|x_k - x^*\| \leq \frac{2l}{L}q^{2^k},$$

which completes the proof. □

Now, we show that all the conditions of the theorem are essential and it is generally impossible to strengthen its assertion. Clearly, the existence of a second derivative is required in the formulation of the method, and the strong convexity condition ensures the existence of $[\nabla^2 f(x_k)]^{-1}$. When we drop the Lipschitz condition on $\nabla^2 f(x)$, the weaker requirements for smoothness may diminish the covergence rate of the method. For example [94], let $f(x) = |x|^{\frac{5}{2}}$, $x \in \mathbb{R}$. Then, for $x > 0$, $\nabla f(x) = \frac{5}{2}x^{\frac{3}{2}}$, $\nabla^2 f(x) = \frac{15}{4}x^{\frac{1}{2}}$. Note that $\nabla^2 f(x)$ does not satisfy the Lipschitz condition. The method takes the form, where the initial condition $x_0 > 0$

$$\begin{aligned} x_{k+1} &= x_k - \frac{4}{15}x_k^{-\frac{1}{2}} \cdot \frac{5}{2}x_k^{\frac{3}{2}} \\ &= \frac{1}{3}x_k. \end{aligned}$$

That is, $x_k = \left(\frac{1}{3}\right)^k x_0$ and the method converges to $x^* = 0$ with the rate of geometric progression, rather than quadratically. Finally, it is impossible to assert that the method converges for just any initial approximation, which is not satisfying (4.5). Suppose the problem consists in minimizing the one-dimensional function a derivative of which is shown in Figure 4.1. This function is twice differentiable, strongly convex, $\nabla^2 f(x)$ satisfies a Lipschitz condition and $x^* = 0$. However, if one starts the iterative process from any point x_0 with $|x_0| > 1$, then

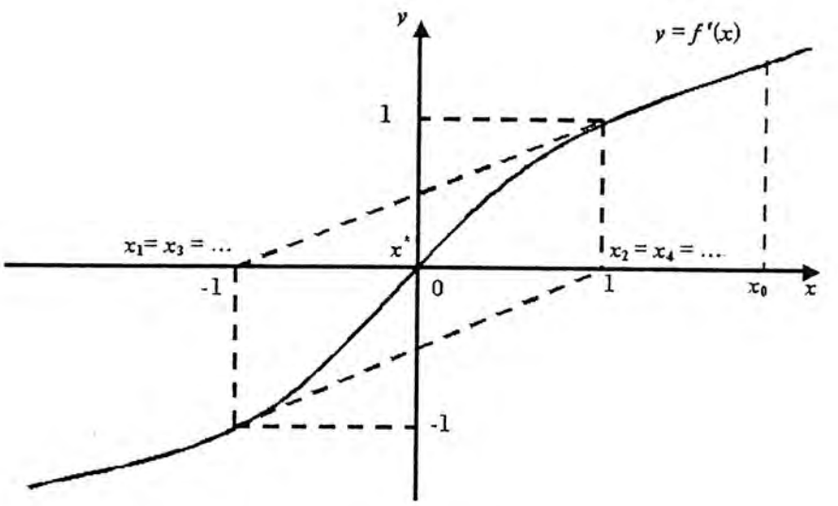

Figure 4.1: Divergence of Newton's method.

the method does not converge: $|x_k| \equiv 1$, for all $k \geq 1$.

The conditions of Theorem 4.1 can be somewhat relaxed only in one instance: the local conditions in place of the global ones on $f(x)$.

Theorem 4.2 *Assume $f(x)$ is twice differentiable in a neigborhood D of a non-singular minimum point x^*, and $\nabla^2 f(x)$ satisfies a Lipschitz condition on D. Then, there exists an $\varepsilon > 0$ such that for $\|x_0 - x^*\| \leq \varepsilon$, the method (4.2) converges to x^* quadratically.*

For the quadratic function $f(x) = \frac{1}{2}x^T Ax - x^T b$ with A is positive definite, Newton's method converges in only one step, that is, $x_1 = x^*$ for any $x_0 \in \mathbb{R}^n$. This is obvious since the approximating function $\tilde{f}(x)$ coincides with $f(x)$. The closer $f(x)$ is to being quadratic, the faster Newton's method converges. Moreover, if the value of the Lipschitz constant L is smaller, then the domain of

convergence defined by (4.5) is larger, and the convergence rate defined by the quantity q is also faster.

4.2.3 Newton's Method with Steplength

We should note that, if the initial guess x_0 is too far away from the minimizer x^*, then $G_k := \nabla^2 f(x_k)$ may not be positive definite. Therefore, Newton's direction may not be the descent direction, so it may not converge. Hence, taking the steplength $\alpha_k = 1$ is not always suitable. We should use the line search method for the Newton's method to determine the steplength α_k. But it should be emphasized that, only when the steplength α_k converges to 1, the convergence rate of Newton's method can be quadratic.

Now, the iterative formula for the Newton's methods becomes [128]:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{4.8}$$

$$d_k = -(G_k)^{-1}\nabla f(x_k) \tag{4.9}$$

where α_k is the steplength found by the line search.

Algorithm 4.2 (Newton's Method with line search)

Step 1: Given $x_0 \in \mathbb{R}^n$, $\epsilon > 0$ *and* $k := 0$.

Step 2: Calculate $g = \nabla f(x)$ *and* $G = \nabla^2 f(x)$.

Step 3: If $\nabla f(x_k) \geq \varepsilon$, *then go to Step 4; otherwise, stop.*

Step 4: Solve the linear system for the Newton's direction d_k:

$$G_k d = -g_k.$$

Step 5: Conduct the line search, find α_k *which satisfies*

$$f(x_k + \alpha_k d_k) = \min_{\alpha > 0} f(x_k + \alpha d_k).$$

Step 6: Calculate $x_{k+1} = x_k - \alpha_k d_k$.

Step 7: Set $k := k + 1$, *then go back to Step 3.*

4.2.4 Convergence Analysis of Newton's Method with Steplength

In this subsection, we follow [128] to show that Newton's method with line search converges globally.

Theorem 4.3 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice differentiable on the convex set $D \subset \mathbb{R}^n$. Suppose for all $x_0 \in D$, there exists a constant $m > 0$ such that $f(x)$ satisfies*

$$u^T \nabla^2 f(x) u \geq m\|u\|^2, \forall u \in \mathbb{R}^n, x \in L(x_0), \tag{4.10}$$

where $L(x_0) = \{x | f(x) \leq f(x_0)\}$. Then, by the exact linear line search, the iterative point $\{x_k\}$ produced by the Newton's method with steplength satisfies the following conditions:

(1) If $\{x_k\}$ is a finite sequence, then $\nabla f(x_k) = 0$ for some k.

(2) If $\{x_k\}$ is a infinite sequence, then $\{x_k\}$ must converge to a unique minimizer x^*.

Proof: First, from (4.10), we know that the function $f(x)$ is strictly convex in $\mathbb{R}^n$. So, the stationary point of $f(x)$ should be the minimizer, which is unique.

By assumption, note that the set $L(x_0)$ is bounded, closed and convex. Since $f(x_k)$ is strictly decreasing, so we know that $(x_k) \subset L(x_0)$, and (x_k) is bounded. Then, there exists a limit point $\bar{x} \in L(x_0)$ such that $x_k \to \bar{x}$. Also note that $f(x_k)$ is strictly decreasing and bounded below, then $f(x_k) \to f(\bar{x})$. Therefore, by Theorem 3.4, we have $g_k \to g(\bar{x}) = 0$. Again, since the stationary point is unique, then the sequence (x_k) must converge to $\bar{x}$. □

Similarly, if the linear line search satisfies

$$f(x_k) - f(x_k + \alpha_k d_k) \geq \bar{\eta} \|g_k\|^2 \cos^2\langle d_k, -g_k\rangle, \tag{4.11}$$

where $\bar{\eta}$ is a constant independent of k. Then, the convergence still holds.

Theorem 4.4 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice differentiable on the convex set $D \subset \mathbb{R}^n$. Suppose for all $x_0 \in D$, there exists a constant $m > 0$ such that $f(x)$ satisfies (4.10) on the set $L(x_0)$. Then, under the condition (4.11) of the linear line search, the iterative point $\{x_k\}$ which is produced by Newton's method satisfies*

$$\lim_{k\to\infty} \|\nabla f(x_k)\| = 0, \tag{4.12}$$

and $\{x_k\}$ converges to the unique minimizer of $f(x)$.

Proof: Since $f(x)$ satisfies (4.10), then $f(x)$ is uniformly convex on the set $L(x_0)$. From (4.11), we know that $f(x)$ is strictly decreasing, so the sequence (x_k) is bounded. Hence, there exists a constant $M > 0$ such that

$$\|\nabla^2 f(x_k)\| \leq M \tag{4.13}$$

for all k. By the iterative formula of Newton's method, (4.10) and (4.13), we have:

$$\begin{aligned} \cos\langle d_k, -g_k\rangle &= \frac{-d_k^T g_k}{\|d_k\|\|g_k\|} \\ &= \frac{g_k^T \nabla^2 f(x_k)^{-1} g_k}{\|\nabla^2 f(x_k)^{-1} g_k\|\|g_k\|} \\ &= \frac{d_k^T \nabla^2 f(x_k) d_k}{\|d_k\|\|\nabla^2 f(x_k) d_k\|} \\ &\geq \frac{m}{M}. \end{aligned} \tag{4.14}$$

Hence, we have:

$$\infty > \sum_{k=0}^{\infty} [f(x_k) - f(x_{k+1})] \geq \sum_{k=0}^{\infty} \bar{\eta} \frac{m^2}{M^2} \|g_k\|^2. \tag{4.15}$$

Then, we can have the statement (4.12). Note that $f(x)$ is strictly convex, so there is just one stationary point, and from (4.12), the sequence (x_k) converges to the unique minimizer x^* of $f(x)$. □

4.3 Greenstadt's Method

It is possible that $G_k := \nabla^2 f(x_k)$ is indefinite or singular. To improve the above direction of search, Greenstadt [62] suggested the following variant of the Newton's method:

Let $(\lambda_k)_j$ be the jth eigenvalue of G_k and $(v_k)_j$ be its corresponding eigenvector such that $(v_k)_j^T(v_k)_j = 1$.

Now, we define a $n \times n$ matrix $\bar{G}_k$ as

$$\bar{G}_k := \sum_{j=1}^{n} |(\lambda_k)_j|(v_k)_j(v_k)_j^T. \tag{4.16}$$

The direction of search in the kth iteration of Newton's method is replaced by

$$x_k = x_{k-1} - (\bar{G}_{k-1})^{-1}\nabla f(x_{k-1}). \tag{4.17}$$

This is still inadequate definition of the direction of search since $\bar{G}_k$ could be singular. To avoid this difficulty, we define $\bar{G}_k$ as follows:

$$\bar{G}_k := \sum_{j=1}^{n} \beta_j(v_k)_j(v_k)_j^T, \tag{4.18}$$

where

$$\beta_j = \max\{|(\lambda_k)_j|, \delta\}.$$

The positive scalar δ is machine dependent. If a machine has a t bit word length, then the usual choice for δ is

$$\delta = 2^{-\frac{t}{2}}.$$

The matrix $\bar{G}_k$ is positive definite so this variant of Newton's method is a descent algorithm under the norm

$$\|y\|^2 = y^T \bar{G}_k y.$$

It follows from the definition of $\bar{G}_k$ that

$$(\bar{G}_k)^{-1} = \sum_{j=1}^{n} \beta_j^{-1} (v_k)_j (v_k)_j^T, \tag{4.19}$$

consequently the direction vector $d_k := (\bar{G}_k)^{-1} \nabla f(x_k)$ can be calculated in only a further $O(n^2)$ operations once the $(\lambda_k)_j$ and $(v_k)_j$ are known. The method is satisfactory although the amount of work necessary to compute the eigenvalues and eigenvectors is high. This leads us to suggest the following scheme that avoids unnecessary calculation of the eigenvalues and eigenvectors.

Now, if G_k is positive definite, then the only work necessary to determine d_k is to solve the equations

$$G_k d_k = -\nabla f(x_k). \tag{4.20}$$

The most effective way of doing this is to factorize G_k, suggested by Wilkinson [121]. By using the method of Cholesky, we have

$$G_k = LDL^T, \tag{4.21}$$

where L a lower-triangular matrix with unit-diagonal elements and D is a diagonal matrix. This method requires $\frac{1}{6}n^3 + O(n^2)$ multiplications compared with a conservation estimate of $3n^3$ multiplications for the eigenvector analysis. If G_k is not positive definite, then this will be revealed in the factorization by the occurrence of a non-positive diagonal in D.

Algorithm 4.3 (Greenstadt's Method)

Step 1: Given $x_0 \in \mathbb{R}^n$ *and* $k := 0$.

Step 2: Calculate $g_k = \nabla f(x_k)$ *and* $G_k = \nabla^2 f(x_k)$.

Step 3: Find the eigenvalues $(\lambda_k)_{j=1}^n$ *and its corresponding eigenvectors* $(v_k)_{j=1}^n$, *with* $(v_k)_j^T (v_k)_j = 1$, *of the matrix* G_k.

Step 4: Find $\bar{G}_k = \sum_{j=1}^n \beta_j (v_k)_j (v_k)_j^T$, *where* $\beta_j = \max\{|(\lambda_k)_j|, \delta\}$, *for some* $\delta > 0$.

Step 5: Calculate $x_{k+1} = x_k - (\bar{G}_k)^{-1} \nabla f(x_k)$.

Step 6: Set $k := k + 1$, *then go back to Step 2.*

4.4 Marquardt-Levenberg Method

This method is an adaption of the Marquardt-Levenberg algorithm for the solution of non-linear least-squares problems ([79], [81], [84], [122]). In this method, the direction of search is given by

$$d_k = -(\bar{G}_k)^{-1} g_k, \tag{4.22}$$

where

$$\bar{G}_k = G_k + \beta_k Q_k, \tag{4.23}$$

and β_k is a non-negative scalar and Q_k is some specified matrix which is at least positive semi-definite. There are some suggestions for the matrix Q_k, for example:

(i) $Q_k = I$, the unit matrix;

(ii) $Q_k = D_k$, the diagonal matrix whose elements are the absolute value of the diagonal elements of G_k.

The scalar α_k is taken to be unity while β_k is chosen such that $f(x_{k+1}) < f(x_k)$ and the matrix $\bar{G}_k$ is a positive-definite matrix. This method generates the descent method under the norm

$$\|y\|^2 = y^T \bar{G}_k y.$$

There is a disadvantage of this method. Since a suitable β_k is not known initially, so for each estimation of β_k, a new set of linear system of equations must be solved in order to determine the corresponding d_k.

But when solving the non-linear least-squares problem by this method using $Q_k = I$, there is a suggested method to avoid this difficulty [6]:

Let λ_j be the jth eigenvalue of the matrix G_k and v_j be its corresponding eigenvector, where $v_j^T v_j = 1$. Then,

$$(G_k)^{-1} = \sum_{j=1}^{n} \lambda_j^{-1} v_j v_j^T, \tag{4.24}$$

and

$$(\bar{G}_k)^{-1} = \sum_{j=1}^{n} (\lambda_j + \beta_k)^{-1} v_j v_j^T. \tag{4.25}$$

Hence, once if the eigenvalues λ_j and the eigenvectors v_j are known, then each of the trial, the directions d_k can be determined in $O(n^2)$ operations.

The mathematicians examine some relative amounts of work required for solving sets of linear systems of equations and computing eigenvalues, shows that this scheme would only be worthwhile if 20 or more trial β_k's were required. Each trial β_k has an associated function evaluation, the test of a satisfactory β_k being that this function value is less than the current lowest function value. If the algorithm required 20 function evaluations per iteration, it would be judged very

unsatisfactory. A further disadvantage is that storage of the eigenvectors requires twice number of locations as that required for the Cholesky factors.

Algorithm 4.4 (Marquardt-Levenberg Method)

Step 1: Given $x_0 \in \mathbb{R}^n$ *and* $k := 0$.

Step 2: Calculate $g_k = \nabla f(x_k)$ *and* $G_k = \nabla^2 f(x_k)$.

Step 3: Find the eigenvalues $(\lambda_k)_{j=1}^n$ *and its corresponding eigenvectors* $(v_k)_{j=1}^n$, *with* $(v_k)_j^T (v_k)_j = 1$, *of the matrix* G_k.

Step 4: Determine the value of β_k *such that* $f(x_k) < f(x_{k-1})$ *and the matrix* $\bar{G}_k := G_k + \beta_k I$ *is positive definite.*

Step 5: Find $(\bar{G}_k)^{-1} = \sum_{j=1}^n (\lambda_j + \beta_k)^{-1} (v_k)_j (v_k)_j^T$.

Step 6: Calculate $x_{k+1} = x_k - (\bar{G}_k)^{-1} \nabla f(x_k)$.

Step 7: Set $k := k + 1$, *then go back to Step 2.*

4.5 Fiacco and McComick Method

Follow from [86], we recall that if A is an $n \times n$ symmetric and positive-definite matrix, it can be factorized in the following form:

$$A = LDL^T, \tag{4.26}$$

where L is a lower triangular matrix with unit diagonal elements and D is a diagonal matrix whose diagonal elements are all positive.

This factorization (4.26) is called Cholesky factorization of A and the matrices L and D are the Cholesky factors of A. The elements of L and D are unique, and they can be determined either column by column or row by row from comparing

the elements in (4.26). Suppose the first $k-1$ columns of L and D have been determined, then the kth columns can be determined from the following:

Let $l_{i,j}$ and $a_{i,j}$ denote the (i,j)th element of L and A respectively. Let $d_{j,j}$ be the jth diagonal element of D. Comparing the (k,k)th element in (4.26), we have:

$$\sum_{i=1}^{k} d_{i,i} l_{k,i}^2 = a_{k,k}, \tag{4.27}$$

Hence, we have:

$$d_{k,k} = a_{k,k} - \sum_{i=1}^{k-1} d_{i,i} l_{k,i}^2. \tag{4.28}$$

By comparing the kth column of A, we have:

$$\sum_{i=1}^{k} d_{i,i} l_{j,i} l_{k,i} = a_{j,k}, \tag{4.29}$$

for $j = k+1, ..., n$. Then, we have:

$$l_{j,k} = \frac{a_{j,k} - \sum_{i=1}^{k-1} d_i l_{j,i} l_{k,i}}{d_{k,k}}, \tag{4.30}$$

for $j = k+1, ..., n$.

It may happen that the value of $d_{k,k}$ is very small, but it follows from the relationship (4.27) that this does not result in a large element in $LD^{\frac{1}{2}}$ expect possibly when A has a large diagonal element. The numerical stability of this factorization method is a direct result of the priori bounds on the elements of $LD^{\frac{1}{2}}$ given by (4.27). If A is not positive definite matrix, then the factorization (4.26) does not exist. If there is such a factorization, then at least one of the elements of D is negative, and (4.27) can no longer be uesd to give the priori bounds on the elements of $LD^{\frac{1}{2}}$.

The method of Fiacco and McCormick is given as follows ([43], [41], [42]): at the kth iteration, we factorize the Hessian matrix G_k:

$$G_k = L_k D_k L_k^T. \tag{4.31}$$

If the diagonal elements of D_k are all positive, then the direction of search d_k can be determined by the Newton's method, that is:

$$d_k = -(G_k)^{-1} g_k. \tag{4.32}$$

Now, we have the Cholesky factorization of G_k, then:

$$\begin{aligned} (G_k)^{-1} &= (L_k D_k L_k^T)^{-1} \\ &= (L_k^T)^{-1} \tilde{D}_k (L_k)^{-1}, \end{aligned}$$

where

$$\tilde{D}_k = (1/d_{i,i})_{i=1}^n.$$

On the other hand, if some diagonal elements of D_k are zero or negative, we should find some other methods to obtain the direction of search d_k. But it will involve the use of L_k and D_k, and it cannot be relied upon [121]. So, d_k may now be an arbitrary descent direction in no way related to G_k.

Algorithm 4.5 (Fiacco and McCormick Method)

Step 1: Given $x_0 \in \mathbb{R}^n$ *and* $k := 0$.

Step 2: Calculate $g_k = \nabla f(x_k)$ *and* $G_k = \nabla^2 f(x_k)$.

Step 3: If G_k *is positive definite, then factorize into* $G_k = L_k D_k L_k^T$.

Step 4: Calculate $x_{k+1} = x_k - (G_k)^{-1} \nabla f(x_k)$.

Step 5: Set $k := k + 1$, *then go back to Step 2.*

4.6 Matthews and Davies Method

This method is almost the same as the method of Fiacco and McCormick in Section 4.5, but this time we factorize the matrix G_k in the kth iteration into the following form:

$$G_k = L_k U_k, \tag{4.33}$$

where L_k is a lower triangular matrix with unit diagonal elements and U_k is an upper triangular matrix.

Matthews and Davies, who described this method in 1969 [82] and slightly modified this in 1971 [83], suggested that the above factorization should be performed by Gaussian elimination. This method is similar to Cholesky factorization, but twice as long and uses twice the storage. It is clear from assumptions that they make about the factorization that in their implementation of Gaussian elimination, they pivot down the diagonal. If G_k is positive definite, then the matrix L_k is the same to that given by Cholesky factorization and

$$U_k = D_k L_k^T.$$

In the process of Gaussian elimination, there is some more arrangements in the diagonal elements of U_k as the following:

$$u_{i,i} = \begin{cases} u_{i,i}, & \text{if } u_{i,i} > 0; \\ 1, & \text{if } u_{i,i} = 0; \\ -u_{i,i}, & \text{if } u_{i,i} < 0. \end{cases}$$

This does not affect the process if G_k is positive definite since all the diagonal elements of U_k must be positive. On the other hand, if G_k is not positive definite, then it does not guarantee that the elements in U_k and L_k become very large in

the above modifications. Hence, this consequences are identical to the method of Fiacco and McCormick.

Algorithm 4.6 (Matthews and Davies Method)

Step 1: *Given* $x_0 \in \mathbb{R}^n$ *and* $k := 0$.

Step 2: *Calculate* $g_k = \nabla f(x_k)$ *and* $G_k = \nabla^2 f(x_k)$.

Step 3: *Factorize* G_k *into* $G_k = L_k U_k$.

Step 4: *If the diagonal elements of* U_k, *that is* $u_{i,i} \neq 0$, *then set* $u_{i,i} = |u_{i,i}|$; *otherwise, set* $u_{i,i} = 1$.

Step 5: *Calculate* $x_{k+1} = x_k - (G_k)^{-1} \nabla f(x_k)$.

Step 6: *Set* $k := k + 1$, *then go back to Step 2.*

4.7 Numerically Stable Modified Newton's Method

In the above two factorization methods to factorize G_k, when we solve the equation

$$G_k d_k = -\nabla f(x_k),$$

for the direction of search d_k, it seems that they are both numerically unstable. So, we should find a modified method which is numerically stable for the iterative process. We now follow [86] to present some results in this aspect.

Recall that a matrix A can be factorized into the form

$$A = LDL^T. \tag{4.34}$$

If A is positive definite, we can also have the factorization [60], [114]:

$$A = \bar{L}\bar{L}^T, \tag{4.35}$$

where $\bar{L}$ is a lower triangular matrix. The relationship between L and $\bar{L}$ is

$$\bar{L} = LD^{\frac{1}{2}}.$$

Note that the factorization (4.34) is preferred to (4.35) since it avoids the need to form the square roots of the elements of D. In the modifications to the factorization process in this section, the square roots occur irrespective of which factorization is used. Hence, we use the factorization (4.35) since this simplifies the description.

Let $\bar{l}_{i,j}$ and $a_{i,j}$ denote the (i,j)th element of $\bar{L}$ and A respectively. Comparing the (k,k)th element in (4.35), we have:

$$\sum_{i=1}^{k} \bar{l}_{k,i}^2 = a_{k,k}, \tag{4.36}$$

Hence, we have:

$$\bar{l}_{k,k} = \left(a_{k,k} - \sum_{i=1}^{k-1} \bar{l}_{k,i}^2 \right)^{\frac{1}{2}}. \tag{4.37}$$

By comparing the kth column of A, we have:

$$\sum_{i=1}^{k} \bar{l}_{j,i}\bar{l}_{k,i} = a_{j,k}, \tag{4.38}$$

for $j = k+1, ..., n$. Then, we have:

$$\bar{l}_{j,k} = \frac{a_{j,k} - \sum_{i=1}^{k-1} \bar{l}_{j,i}\bar{l}_{k,i}}{\bar{l}_{k,k}}, \tag{4.39}$$

for $j = k+1, ..., n$.

Hence, in the algorithm, when we find the direction of search, it is equivalent to solve the equation:

$$\bar{L}_k \bar{L}_k^T d_k = -\nabla f(x_k), \tag{4.40}$$

where $\bar{L}_k$ is a lower triangular matrix.

The matrix $\bar{L}_k$ is determined by applying the Cholesky's method to the matrix G_k. The resulting factorization is not necessarily the same as G_k, but some of the other matrix, say $\bar{G}_k$. The deviations from Cholesky's method occur when tests carried out during the process of factorizing G_k indicate that G_k is not positive definite. The main objective of the above changes is to make sure that $\bar{G}_k$ differs from G_k in some minimum way, but $\bar{G}_k$ is positive definite.

Suppose the modified factorization procedure is applied to a symmetric but indefinite matrix A, whose smallest eigenvalue is λ. Let the resulting factor be $\bar{L}$. The prerequisite for A to differ from $\bar{L}\bar{L}^T$ in some minimum way is for the following quantity:

$$\min_{\bar{L}} \|A - \bar{L}\bar{L}^T\|.$$

This quantity is a continuous function of λ, which is equal to 0 when $\lambda = 0$. In fact, it is not easy to achieve this since there is a need to bound the condition number of $\bar{G}_k$ in order to avoid the numerical difficulties in the evaluation of d_k. Hence, in this modification

$$\|G_k - \bar{G}_k\|$$

tends uniformly to 0 as λ tends to δ, where $\delta > 0$ is a small scalar to the word length of the computer employed.

There is a very important property lost when the matrix is not positive definite or very close to singular in the Cholesky's factorization. This property is

that the priori bound on the elements of the lower triangular matrix $\bar{L}$ given by (4.36). In the modification, this property is not required since the procedure acts directly to limit the size of the elements. It is clear from (4.39) that the elements of the off-diagonal from the matrix $\bar{L}$ can be always reduced, if they were too large, by increasing the diagonal element of $\bar{L}$.

If the factorization is performed row by row, then it is not possible to bound the change necessary in the jth diagonal element in order to bound the remaining elements in the jth row. It will be shown that it is possible to bound the change in the diagonal elements if the factorization is performed column by column.

To simplify the description of this modification, the superfix k will be temporarily dropped. Consider the ith column of $\bar{L}$. Given the first $i-1$ columns of $\bar{L}$, we can determine the ith column and that

$$|\bar{l}_{r,s}| \leq \beta, \tag{4.41}$$

where $r = 2, ..., n$, $s = 1, ..., r-1$ and for some fixed scalar $\beta > 0$.

Define

$$\hat{l}_i = \max\left\{\delta, |G_{i,i} - \sum_{r=1}^{i-1} \bar{l}_{i,r}^2|^{\frac{1}{2}}\right\} \tag{4.42}$$

and

$$\hat{l}_j = \frac{G_{i,j} - \sum_{r=1}^{i-1} \bar{l}_{i,r}\bar{l}_{j,r}}{\bar{l}_i}, \tag{4.43}$$

where $j = i+1, ..., n$ and $\delta > 0$ is some small scalar dependent on the word-length of the computer being employed. A suitable choice of this parameter on a computer with a t bit word-length is

$$\delta = 2^{-\frac{t}{2}}.$$

Let

$$\theta = \max\left\{|\hat{l}_j| : j = i+1, ..., n\right\}.$$

If $\theta \leq \beta$, then

$$\bar{l}_{j,i} = \hat{l}_j, \tag{4.44}$$

for $j = i, ..., n$; otherwise, if $\theta > \beta$, then

$$\bar{l}_{i,i} = \frac{\theta \hat{l}_i}{\beta} \tag{4.45}$$

and

$$\bar{l}_{j,i} = \frac{\beta \hat{l}_j}{\theta}, \tag{4.46}$$

where $j = i+1, ..., n$.

It is clear that no matter which definitions is used, we have:

$$|\bar{l}_{j,i}| \leq \beta,$$

where $j = i+1, ..., n$.

Now, we show that all diagonal elements of $\bar{L}$ are bounded. First, by (4.42), we have:

$$\begin{aligned} \hat{l}_i &\leq \left(|G_{i,i}| + \sum_{r=1}^{i-1} \bar{l}_{i,r}^2\right)^{\frac{1}{2}} + \delta \\ &\leq \left(|G_{i,i}| + \sum_{r=1}^{i-1} \beta^2\right)^{\frac{1}{2}} + \delta, \end{aligned}$$

which implies

$$\hat{l}_i \leq \left(|G_{i,i}| + (i-1)\beta^2\right)^{\frac{1}{2}} + \delta. \tag{4.47}$$

Also, from the statement (4.43), we have:

$$|\hat{l}_j| \leq \theta \leq \frac{\xi_i + (i-1)\beta^2}{\hat{l}_i}, \tag{4.48}$$

where

$$\xi_i = \max\{|G_{i,j}| : j = i+1, ..., n\}.$$

Note that if $\theta \leq \beta$, from (4.44), we have $\bar{l}_{i,i} = \hat{l}_i$, then from the inequality (4.47), $\bar{l}_{i,i}$ is bounded.

Also, if $\theta > \beta$, from (4.45) and (4.48), we have:

$$\bar{l}_{i,i} \leq \frac{\xi_i + (i-1)\beta^2}{\beta},$$

which implies

$$\bar{l}_{i,i} \leq \frac{\xi_i}{\beta} + (i-1)\beta. \tag{4.49}$$

This completes the proof of all diagonal elements of $\bar{L}$ are bounded.

There are two criterions for the choice of the parameter β. First, the value of β should be large enough so that G is sufficiently positive definite for $\bar{G} = G$. Also, we wish to minimize the bound (4.49) for all i.

Now, from (4.36), if

$$\beta^2 \geq \max_i\{|G_{i,i}|\} := \gamma,$$

then the first criterion is satisfied. Also, the bound on the diagonal elements will be minimized if

$$\beta^2 = \frac{\xi}{\gamma},$$

where

$$\xi = \max_i\{\xi_i\}.$$

Hence, the parameter β is defined to be the following:

$$\beta = \max\left\{\gamma^{\frac{1}{2}}, \left(\frac{\xi}{n}\right)^{\frac{1}{2}}\right\}. \tag{4.50}$$

Therefore, from (4.49) and (5.17), we have:

$$\delta \leq \bar{l}_{i,i} \leq 2n\beta, \tag{4.51}$$

for $i = 1, ..., n$.

It is important to realize that the definition of the elements of the off-diagonal matrix given by (4.46) is identical to that which would have resulted if the diagonal elements had been given by (4.45) in applying Cholesky's factorization. The lower triangular matrix obtained by this procedure is therefore identical to that which would have been obtained by applying Cholesky's method to the matrix

$$\bar{G} = G + D, \tag{4.52}$$

where D is a diagonal matrix.

Clearly, the elements of D are bounded. Then, from (4.52), the ith element of D, say d_i, is given by:

$$d_i = \sum_{j=1}^{i} \bar{l}_{i,j} - G_{i,i}. \tag{4.53}$$

Note that if

$$\bar{l}_{i,i} = \left(G_{i,i} - \sum_{j=1}^{i-1} \bar{l}_{i,j}\right)^{\frac{1}{2}},$$

then

$$d_i = 0.$$

The number of operations taken by the procedure is $\frac{1}{6}n^2 + O(n^2)$ and the additional storage required for the matrix G can be negligible since $\bar{L}$ can be

overwritten on G.

Now, let us consider again the kth iteration. Note that the direction of search in the kth iteration is found by solving the following linear equation:

$$\bar{L}_k \bar{L}_k^T d_k = -\nabla f(x_k),$$

and the next iterative point x_{k+1} is given by:

$$x_{k+1} = x_k + \alpha_k d_k.$$

Since $\bar{G}_k$ is positive definite, then d_k is the descent direction direction under the norm

$$\|y\|^2 = y^T(G_k + D_k)y.$$

There is another procedure to avoid the need to determine the direction of search d_k. We define d_k to be the descent direction under the norm

$$\|y\|^2 = y^T(\bar{L}_k + \lambda\hat{L})(\bar{L}_k + \lambda\hat{L})^T y, \tag{4.54}$$

where $\hat{L}$ is a specified lower triangular matrix and λ is a non-negative scalar chosen so that

$$f(x_{k+1}) < f(x_k). \tag{4.55}$$

There are some possible choices of the lower triangular matrix $\hat{L}$, for example:

(1) I, the identity matrix;

(2) L_{k-1}, the lower triangular matrix in the Cholesky's factorization of G_{k-1} in the $(k-1)$th iteration;

(3) D_k, the diagonal matrix in the Cholesky's factorization of G_k in the kth iteration;

(4) $\hat{D}$, the diagonal matrix whose ith diagonal element is $\bar{l}_{i,i}$.

When we choose $\hat{L} = L_{k-1}$ in (2), there are two disadvantages about this. First, the search direction d_k may turn to the previous ones. So, it may not be the descent direction under the norm (4.54). Also, when we compute the kth iteration, it needs to store an additional lower triangular matrix L_{k-1} at the previous iteration. But the other three choices of $\hat{L}$ should work and satisfy the condition (4.55) if the value of λ is chosen large enough. The above procedure is similar to the Marquardt-Levenberg method in Section 4.4, but this has some additional advantages. First, we can avoid the need to repeatedly solve a system of linear equations. Second, it uses less storage for the other additional matrices. Further, it enables a better choice to be made of the weighting matrix.

Algorithm 4.7 (Numerically Stable Modified Newton's Method [86])

Step 1: *Given* $x_0 \in \mathbb{R}^n$ *and* $k := 0$.

Step 2: *Calculate* $g_k = \nabla f(x_k)$ *and* $G_k = \nabla^2 f(x_k)$.

Step 3: *Factorize* G_k *into* $G_k = \bar{L}_k \bar{L}_k^T$.

Step 4: *Find the values of* $\gamma = \max_i\{|G_{i,i}|\}$ *and* $\xi = \max_{i,j}\{|G_{i,j}|\}$.

Step 5: *Find the value of* $\beta = \max\{\gamma^{\frac{1}{2}}, \left(\frac{\xi}{n}\right)^{\frac{1}{2}}\}$.

Step 6: *From the* i*th column of* $\bar{L}$, *find* $\hat{l}_i = \max\{\delta, |G_{i,i} - \sum_{r=1}^{i-1} \bar{l}_{i,r}^2|^{\frac{1}{2}}\}$ *and* $\hat{l}_j = \frac{G_{i,j} - \sum_{r=1}^{i-1} \bar{l}_{i,r}\bar{l}_{j,r}}{\hat{l}_i}$, *for some* $\delta > 0$.

Step 7: *Find* $\theta = \max\{|\hat{l}_j| : j = i+1, ..., n\}$.

Step 8: *From Step 5 and 7, if* $\theta \leq \beta$, *then set* $\bar{l}_{j,i} = \hat{l}_j$, *for* $j = i, ..., n$; *otherwise, set* $\bar{l}_{i,i} = \frac{\theta \hat{l}_i}{\beta}$ *and* $\bar{l}_{j,i} = \frac{\beta \hat{l}_j}{\theta}$, *for* $j = i+1, ..., n$.

Step 9: *Calculate* $x_{k+1} = x_k - \left\{\left(\bar{L}_k + \lambda\hat{L}\right)\left(\bar{L}_k + \lambda\hat{L}\right)^T\right\}^{-1} \nabla f(x_k)$, *for some lower triangular matrix* $\hat{L}$ *and* $\lambda \geq 0$ *satisfies* $f(x_{k+1}) < f(x_k)$.

Step 10: Set $k := k + 1$, then go back to Step 2.

4.8 The Role of the Second Derivative Methods

This section follows [86] to discuss some roles of some second derivative methods. There is an important advantage for using the second derivative methods. They are able to distinguish between most of the saddle points and the local minimizers. The exceptions are those saddle points where the Hessian matrix has negative eigenvalues, but if the eigenvalues are so close to zero, the computer is unable to distinguish them from positive eigenvalues. Even in these circumstances, it is very unlikely that the procedures described would converge to such a point.

On the other hand, the line search in a second derivative method does not play such a critical role as a line search does in most first derivative methods. In the methods that requires a linear search a step that merely reduces the function will suffice. However, another feature for the second derivative method is that the initial predicted step to a minimum along a search direction is much more accurate than that given by first derivative methods. These two features result in the number of function evaluations per iteration being smaller for second derivative methods.

A question still to be answered is when should a second derivative method be used. This will depend on the following factors, most of which are unknown:

(1) T_1 = the time(s) to evaluate f;

(2) T_2 = the time(s) to evaluate $g = \nabla f$;

(3) T_3 = the time(s) to evaluate $G = \nabla^2 f$;

(4) T_4 = the time(s) to execute one step of the algorithm knowing the function and its derivatives.

The ratio r of the total times to minimize a function compared with a first derivative method is given by:

$$r = \frac{n_1 n_3 T_3 + n_2 n_3 (T_1 + T_3) + n_3 T_4}{n_4 n_5 (T_1 + T_3) + n_5 T_5} \tag{4.56}$$

where T_5 is the time(s) to execute one step of a first derivative method given the function and gradient. Define:

(1) n_1 = the frequency the Hessian matrix is evaluated;

(2) n_2 = the average number of functions evaluations per iteration for second derivative method;

(3) n_3 = the number of iterations taken by second derivative method;

(4) n_4 = the average number of function evaluations per iteration for first derivative method;

(5) n_5 = the number of iterations taken by first derivative method.

As the algorithms have been described $n_1 = 1$. If the initial estimation to the solution is poor, then the expected savings in a reduced number of iterations may not be very high. To overcome this problem, the Hessian matrix G need not be evaluated at each iteration. The frequency of evaluation can be related to the lack of progress using the old Hessian matrix and the size of the gradient. If the method in Section 4.7 is used and compared with a rank 2 quasi-Newton's algorithm

$$\frac{T_4}{T_5} \approx \frac{1}{6} \cdot \frac{n^3}{3n^2} = \frac{n}{18},$$

when n is large. So, for $n < 20$, then we have:

$$T_4 \approx T_5.$$

If we assume that

$$\frac{n_3}{n_5} \approx \frac{1}{3},$$

then we have:

$$r \approx \frac{n_1 T_3 + n_2(T_1 + T_2) + T_3}{3n_4(T_1 + T_2) + 3T_5}.$$

Comparing with the first derivative method, we should use the second derivative method if

$$r < 1,$$

that is:

$$T_3 < \frac{1}{n_1}\left[(3n_4 - n_2)(T_1 + T_2) + 3T_5 - T_4\right]. \tag{4.57}$$

Assume $n_1 = \frac{1}{3}$, $n_2 = 3$, $n_4 = 6$ and $n = 60$. Then, the execution times cancel and (4.57) becomes:

$$T_3 < 45(T_1 + T_2).$$

Obviously, this is a very rough guide. But it does indicate that if second derivatives are available, they almost certainly should be used expect when their computation time is extremely large compared with that for the function and gradient.

Chapter 5

Multi-step Methods

In the pervious two chapters, we have considered two algorithms that are conceptually the simplest: the steepest descent method and Newton's method. There are many other methods for solving the unconstrained minimization problems of differentiable functions $f(x)$. We will describe the most interesting ones, either theoretically or computationally. In this chapter, we will specialize to the following problem:

$$\min_{x \in \mathbb{R}^n} f(x),$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a differentiable function.

In Chapter 3, we have discussed in detail the steepest descent method as following:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k). \tag{5.1}$$

Also in Chapter 4, we discussed Newton's method:

$$x_{k+1} = x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k). \tag{5.2}$$

Although the above two methods are simple and have some advantages, they have some drawbacks:

Method	Advantages	Disadvantages
Steepest Descent	1. Global convergence 2. Relaxed conditions on $f(x)$ 3. Computational simplicity	1. Slow convergence 2. Necessary choice of α
Newton	Rapid convergence	1. Local convergence 2. Rigid conditions on $f(x)$ 3. Large volume of computation

From the above table, the positive and negative features of each method are complementary. Our ideal thinking is to develop a new method combining the best features, eschewing the disadvantages. Although such an ideal solution does not exist, we will describe now some possible steps toward it.

5.1 Background

In the steepest descent method and Newton's method, at each step the information obtained in the previous iterations is not used at all. It is natural to try to take into account the pre-history of the process in order to improve the convergence. The main idea of the methods in which the new approximation depends on the s preceding ones:

$$x_{k+1} = \phi_k(x_k, ..., x_{k-s+1}), \tag{5.3}$$

which are called s-step methods. The steepest descent method and Newton's method are one-step methods. Following [80] and [39], we will consider some

multi-step methods, where $s > 1$.

5.2 Heavy Ball Method

One of the simplest multi-step methods is the two-step heavy-ball method [92]:

$$x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta(x_k - x_{k-1}), \tag{5.4}$$

where $\alpha > 0$ and $\beta \geq 0$ are parameters. It is clear that when $\beta = 0$, the method in (5.4) is the original the steepest descent method.

Why the method in (5.4) is called heavy ball method? It comes from the following physical analogy. The motion of a body (or the "heavy ball") in a potential field under the force of friction (or viscosity) is described by the following second order ordinary differential equation:

$$\mu \frac{d^2x}{dt^2} = -\nabla f(x) - p\frac{dx}{dt}, \tag{5.5}$$

where $x = x(t)$ is the function of the motion of the heavy ball at the time t. Clearly, because of energy loss caused by the friction, the body finally reaches a minimum point of the potential function $f(x)$. Hence, the heavy ball solves the corresponding minimization problem. If we consider the difference analog of the equation (5.5), then we have the iterative method (5.4).

Let us consider the term $\beta(x_k - x_{k-1})$. This term is called inertia, which is introduced into the iterative process may increase the rate of convergence. From Figure 5.1, we can see that the motion of the steepest descent method is like "zigzag" shape, but the Heavy Ball method has a smoother trajectory along the "bottom of the gully".

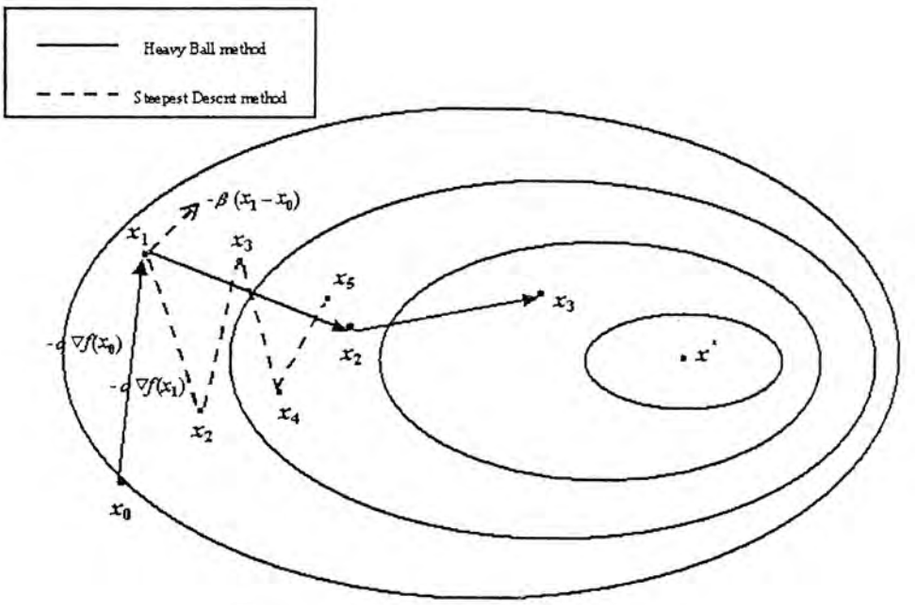

Figure 5.1: The comparison between the Heavy Ball Method and Steepest Descent Method

Let us see the following lemma first [94]:

Lemma 5.1 *Let x^* be a fixed point of*

$$x_{k+1} = g(x_k), \tag{5.6}$$

where g is some mapping from $\mathbb{R}^n$ into $\mathbb{R}^n$ and differentiable function. Let the spectral radius of the Jacobian $g'(x^*)$ satisfy the condition $\rho := \max_{1 \le i \le n} |\lambda_i| < 1$, where λ_i, for $i = 1, ..., n$ are the eigenvalues of $g'(x^*)$. Then, the process (5.1) converges locally linearly to x^* and for every $0 < \varepsilon < 1 - \rho$, we can find a $\delta > 0$ and a constant c such that for all $k \ge 0$

$$\|x_k - x^*\| \le c(\rho + \varepsilon)^k, \tag{5.7}$$

for $\|x_0 - x^*\| \le \delta$.

The above heuristic considerations are strengthened by the following theorem [92]:

Theorem 5.2 *Let x^* be a non-singular minimizer of the function $f(x)$, where $f : \mathbb{R}^n \to \mathbb{R}$. Suppose*

$$\begin{aligned} 0 \leq \beta < 1, \\ 0 < \alpha < \frac{2(1+\beta)}{L}, \quad \textit{and} \\ lI \leq \nabla^2 f(x^*) \leq LI. \end{aligned}$$

Then, we can find an $\varepsilon > 0$ such that for any initial points x_0 and x_1, where $x_0, x_1 \in \mathbb{R}^n$, with $\|x_0 - x^*\| \leq \varepsilon$ and $\|x_1 - x^*\| \leq \varepsilon$, the method in (5.4) converges to the minimizer x^* with the rate of geometric progression:

$$\|x_k - x^*\| \leq c(\delta)(q+\delta)^k, \tag{5.8}$$

where $0 \leq q < 1$ and $0 < \delta < 1 - q$. Moreover, the quantity q is minimal and equal to

$$q^* = \frac{\sqrt{L} - \sqrt{l}}{\sqrt{L} + \sqrt{l}}, \tag{5.9}$$

for

$$\begin{aligned} \alpha^* &= \frac{4}{(\sqrt{L} + \sqrt{l})^2}, \quad \textit{and} \\ \beta^* &= \left(\frac{\sqrt{L} - \sqrt{l}}{\sqrt{L} + \sqrt{l}}\right)^2. \end{aligned}$$

Proof: (*Sketch*)

Here, we cannot apply the procedures for the convergence in this case since they are designed for one-step processes. However, we can increase the dimension of the space which allows us to reduce the multi-step process to a one-step process.

Introduce the $2n$-dimensional vector $z_k = \{x_k - x^*, x_{k-1} - x^*\}$. Then, the iterative process (5.4) can be written in the form

$$z_{k+1} = Az_k + o(z_k), \tag{5.10}$$

where the $2n \times 2n$-square matrix A has the form

$$A = \begin{bmatrix} (1+\beta)I - \alpha B & -\beta I \\ I & 0 \end{bmatrix}, \quad B = \nabla^2 f(x^*). \tag{5.11}$$

Let $l = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n = L$ be the eigenvalue of the matrix B. Then, the eigenvalues ρ_j, $j = 1, ..., 2n$, of the matrix A coincide with the eigenvalues of 2×2-matrix of the form

$$\begin{bmatrix} 1 + \beta - \alpha\lambda_i & -\beta \\ 1 & 0 \end{bmatrix}.$$

Therefore, they are the roots of the equation

$$\rho_2 - \rho(1 + \beta - \alpha\lambda_i) + \beta = 0, \tag{5.12}$$

for $i = 1, ..., n$. One can show that if

$$\begin{aligned} 0 < l &\leq \lambda_i \leq L, \\ 0 \leq \beta &< 1, \\ 0 < \alpha &< \frac{2(1+\beta)}{L}, \end{aligned}$$

then $|\rho| < 1$, where ρ is any root of the equation (5.12).

Now we use Lemma 5.1 to show that the local covergence of iterative processes of the form (5.10), which will allow us to obtain an estimate of (5.8). Calculating

$$\min_{\alpha,\beta} \max_{1 \leq j \leq 2n} |\rho_j|,$$

yields the optimal values of α^*, β^* and the corresponding q^* given in this theorem.

Let us compare now the rate of convergence in the one-step and two-step methods for an optimal choice of parameters. In both cases, we have the geometric rate of convergence, but the progression ratio for the one-step method is equal to

$$q_1 = \frac{L-l}{L+l},$$

whereas for the one-step method, it is equal to

$$q_2 = \frac{\sqrt{L}-\sqrt{l}}{\sqrt{L}+\sqrt{l}}.$$

For large values of the condition number $\mu = \frac{L}{l}$

$$q_1 = 1 - \frac{2}{\mu}, \quad q_2 = 1 - \frac{2}{\sqrt{\mu}}.$$

Hence, to be $e = 2.71828...$ times closer to a solution, the one-step method takes roughly $\frac{\mu}{2}$ iterations, and the two-step method roughly $\frac{\mu}{2}$ iterations. In other words, for ill-posed problems the heavy-ball method yields a roughly $\sqrt{\mu}$-fold payoff against the gradient method. For large μ this difference is quite large. From the computational viewpoint, the method (5.4) is only slightly more complex than the one-step method. Of course, a choice of optimal values for α and β in (5.4) is not simple: we cannot directly use the formulas (5.9), since the bounds of the spectrum of $\nabla^2 f(x^*)$ (the numbers l and L) are usually unknown. □

Algorithm 5.1 (Heavy Ball Method)

Step 1: Given $x_0 \in \mathbb{R}^n$ *and* $k := 0$.

Step 2: Calculate $\nabla f(x_0)$.

Step 3: Calculate $x_1 = x_0 - \alpha \nabla f(x_0)$, *for some* $\alpha > 0$.

Step 4: Set $k := k + 1$ *and calculate* $\nabla f(x_k)$.

Step 5: Calculate $x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta(x_k - x_{k-1})$,

for some parameters $\alpha > 0$ *and* $\beta \geq 0$.

Step 6: Go back to Step 4.

5.3 Conjugate Gradient Method

5.3.1 Some Types of Conjugate Gradient Method

Let us examine another variant of the two-step method – the conjugate gradient method [65]. The conjugate-gradient method which currently plays a significant role in the theory and practice of optimization, emerged in 1952 in works of Hestenes and Stiefel [66] as a technique for solving systems of linear equations with positive definite matrix.

In the conjugate gradient method, there are two parameters to be determined, which are solved by the following two-dimensional optimization problem [94]:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k) + \beta_k(x_k - x_{k-1}), \tag{5.13}$$

with

$$\begin{aligned} \{\alpha_k, \beta_k\} &= \min_{\alpha,\beta} f(x_{k+1}) \\ &= \min_{\alpha,\beta} f(x_k - \alpha_k \nabla f(x_k) + \beta_k(x_k - x_{k-1})). \end{aligned} \tag{5.14}$$

If the function $f(x)$ is a quadratic function, that is

$$f(x) = \frac{1}{2} x^T A x - x^T b, \tag{5.15}$$

where A is an $n \times n$ positive definite matrix, then this problem can be solved

explicitly by the following formulae ([52], [104]):

$$\alpha_k = \frac{\|g_k\|^2 \cdot d_k^T A d_k - d_k^T g_k \cdot d_k^T A g_k}{g_k^T A g_k \cdot d_k^T A d_k - (d_k^T A g_k)^2}, \tag{5.16}$$

and

$$\beta_k = \frac{\|g_k\|^2 \cdot d_k^T A g_k - d_k^T g_k \cdot g_k^T A g_k}{g_k^T A g_k \cdot d_k^T A d_k - (d_k^T A g_k)^2}, \tag{5.17}$$

where

$$g_k = \nabla f(x_k) = A x_k - b, \tag{5.18}$$

and

$$d_k = x_k - x_{k-1}. \tag{5.19}$$

One might expect that the relationship between methods (5.13) and(5.14) is similar to the steepest descent method. It is even less possible that a two step variant of the steepest descent method (5.13) and (5.14) may provide a substantially faster convergence then the heavy ball method. This is not the case, however, in the quadratic case, method (5.13) and (5.14) with a special choice of d is finite, that is, it yields an exact minimum of the quadratic function in (5.15) in a finite number of iterations.

Let x_0 be an arbitrary initial guess, and let x_1 be the first iterative point obtained by the steepest descent method:

$$x_1 = x_0 - \frac{\|g_0\|^2}{g_0^T A g_0} g_0, \tag{5.20}$$

where

$$g_0 = \nabla f(x_0) = A x_0 - b. \tag{5.21}$$

With the above notations and following [94], we have the following lemma:

Lemma 5.3 *The gradients g_0, g_1, ... in each method of (5.13), (5.18) and (5.21), are pairwise orthogonal. That is,*

$$g_k^T g_i = 0 \tag{5.22}$$

for all $i < k$.

Proof: We use induction on k. Let $g_k^T g_i = 0$, where $0 \leq i < k$, for $k \geq 2$, and $g_i \neq 0$, $i = 0, ..., k$. The orthogonality of g_0, g_1, g_2 follows directly from the definition of the method. Multiplying (5.13) on the left by A yields

$$g_{k+1} = g_k - \alpha_k A g_k + \beta_k (g_k - g_{k-1}).$$

It follows from $g_i \neq 0$ for $i \leq k$ that $\alpha_k \neq 0$. Hence, Ag_k is a linear combination of g_{k+1}, g_k and g_{k-1}, and similarly Ag_i, $i < k$, is a linear combination of g_{i+1}, g_i, g_{i-1}, and by induction, $g_j^T A g_i = 0$, $|i - j| > 1$, $i < k$, $j \leq k$. Therefore,

$$g_i^T g_{k+1} = g_i^T \left(g_k - \alpha_k A g_k + \beta_k (g_k - g_{k-1})\right) = 0$$

for $i = 0, ..., k-2$. It follows directly from the equations (5.13), (5.18) and (5.19) that

$$\begin{aligned} g_k^T g_{k+1} &= 0, \\ d_k^T g_{k+1} &= 0. \end{aligned}$$

Finally, from (5.13), replacing k by $k-1$, we have $d_k = -\alpha_{k-1} g_{k-1} + \beta_{k-1} d_{k-1}$. Applying this relation successively, we obtain that d_k is a linear combination of $g_0, g_1, ..., g_{k-1}$, and g_{k-1} has the coefficient $-\alpha_{k-1} \neq 0$. Hence, it follows from $d_k^T g_{k+1} = 0$, $g_i^T g_{k+1} = 0$, $i \leq k-2$, that $g_{k-1}^T g_{k-1} = 0$. Thus, for all $i \leq k$, one will have $g_i^T g_{k+1} = 0$. □

If g_k vanishes, then x_k is a minimum point of $f(x)$. But note that in $\mathbb{R}^n$, it cannot contain more than n non-zero orthogonal vectors. Hence, $g_k = 0$ for some $k \leq n$. Therefore, we have the following theorem [94]:

Theorem 5.4 *The method (5.13), (5.18) and (5.21) yields a minimum point of the quadratic function*

$$f(x) = \frac{1}{2}x^T Ax - x^T b$$

in no more than n iterations.

There is an important fact [94] for the above lemma and theorem. That is, if L is a subspace of $\mathbb{R}^n$ and $f(x)$ is a convex differentiable function, then

$$a^T \nabla f(x^*) = 0, \quad \text{for all } a \in L$$

if and only if x^* is a minimizer of $f(x)$ on L. Hence, this fact and Lemma 5.3 imply that x_k is a minimizer of the quadratic function $f(x)$ on the subspace passing through x_0 and generated by $g_0, g_1, ..., g_{k-1}$.

The conjugate gradient method is better than the methods mentioned before since the other methods may have the unexpected result. We seek the minimum value of k for the times in succession on 2-dimensional subspaces and find it on the entire k-dimensional subspace. This is an important feature of this method and thus making its finiteness clear.

The directions d_k in the conjugate gradient method satisfy the following relation:

$$d_j^T A d_i = 0, \tag{5.23}$$

for all $i \neq j$.

In fact,

$$d_i = x_i - x_{i-1}.$$

Hence, we have:

$$Ad_i = Ax_i - Ax_{i-1} = g_i - g_{i-1}.$$

On the other hand, since d_k is a linear combination of the vectors $g_0, ..., g_{k-1}$, that is:

$$d_k = \sum_{j=0}^{k-1} \mu_j g_j, \tag{5.24}$$

for some $\mu_j \in \mathbb{R}$. Hence, for $i < k$, from Lemma 5.3, we have:

$$d_k^T A d_k = \left(\sum_{j=0}^{k-1} \mu_j g_j \right)^T \cdot (g_i - g_{i-1}) = 0.$$

The vectors d_i connected by (5.23) are called conjugate, or A-orthogonal. This can be explained the name of this methods, that is, the conjugate linear combinations of successive gradients are constructed.

We can observed that the arbitrary conjugate directions s_i, for $i = 1, ..., n$, then $s_j^T A s_i = 0$, for all $i \neq j$. It allows us to solve the following system of linear equations easily:

$$Ax = b, \tag{5.25}$$

where A is a positive definite matrix.

In fact, we will seek the solution in the form of $x = \sum_{i=1}^{n} \alpha_i s_i$. Substituting it into (5.25), computing the scalar product with s_i, and using the A-orthgonality, we have [104]:

$$\alpha_i = \frac{s_i^T b}{s_i^T A s_i}. \tag{5.26}$$

This solution can be given a recursive form: we take an arbitrary x_0 and construct $x_k = x_{k-1} + \alpha_k s_{k-1}$, where α_k are given by (5.26). Then, $x_n = x^*$ is the solution of (5.25). Since the values of α_k in (5.26) can be determined in different ways, that is:

$$\alpha_k = \min_{\alpha} f(x_{k-1} + \alpha s_k),$$

we can see that the fact that we know the systems of conjugate directions makes it possible to find the minimizer of the quadratic function by means of n one-dimensional minimizations. This is an important result and it will be used repeatedly in what follows in constructing other minimization methods. In the conjugate gradient method, the conjugate directions are not chosen as the above but it is constructed from the recurrence formulae.

When the conjugate gradient method in (5.13) and (5.14) is applied to a non-quadratic functions, we can have the following theorem [94]:

Theorem 5.5 *Let $f(x)$ be a continuously differentiable function and x^* be a non-singular minimum point of $f(x)$. Suppose the set $\{x : f(x) < f(x_0)\}$ is bounded. For any $x_0, x_1 \in \mathbb{R}^n$, then the method in (5.13) and (5.14) satisfy the following properties:*

(i) $\nabla f(x_k) \to 0$ as $k \to \infty$;

(ii) The convergence $x_k \to x^*$ with the rate of geometric progression, that is,

$$\|x_k - x^*\| \leq c(\delta)(q + \delta)^k \tag{5.27}$$

where $0 \leq q < 1$ and $0 < \delta < 1 - q$.

The conjugate gradient method can be given yet another form. Consider the following iterative process [24]:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{5.28}$$

where

$$\begin{aligned} \alpha_k &= \min_{\alpha \geq 0} f(x_k + \alpha d_k), \\ d_k &= -g_k + \beta_k d_{k-1}, \\ \beta_k &= \frac{\|g_k\|^2}{\|g_{k-1}\|^2}, \quad \text{for } k \geq 1, \\ \text{and} \quad \beta_0 &= 0. \end{aligned}$$

with

$$g_k = \nabla f(x_k).$$

The above method is applied to non-quadratic problems and expressed heuristic divinations on the method's efficiency [51]. Then, we have the following lemma [26]:

Lemma 5.6 *For the quadratic function (5.15), using the method (5.13), (5.26), (5.17), (5.20) and the other method (5.28) with the same initial guess x_0, then the sequence of points x_k are the same, for $k \geq 1$.*

The proof of convergence is given by Daniel [26].

Since the vectors d_k in (5.28) and the vectors in (5.19) differ only by (nonzero) scalar factors, while the vectors g_k in (5.28) and (5.19) coincide, the process (5.28) possesses the same properties as (5.13) and (5.19), the vectors d_i are conjugate and the gradients g_i are mutually orthogonal. Lemma 5.6 and Theorem 5.4 imply that the method (5.28) yields a minimizer of the quadratic function (5.25) in $\mathbb{R}^n$ in the number of iterations not larger than n. For the non-quadratic problems, the method (5.28) is simpler than the method (5.13) and (5.14) since it requires solution only of a one-dimensional auxiliary minimization problem, rather than a two-dimensional problem. But it is easy to see that in the non-quadratic case, the finiteness property of this method is lost and this method (5.28) turns into an infinite two-step iterative method.

For the non-quadratic problems, the conjugate gradient method is usually applied in a rather different form, where a restart procedure is introduced: at intervals of time, the step is not made by the method (5.28), but as at the initial

point, that is, according to the gradient. It is most natural to make the restart in terms of the number of iterations equal to the dimension of the space [105]:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{5.29}$$

where

$$\begin{aligned}
\alpha_k &= \min_{\alpha \geq 0} f(x_k + \alpha d_k), \\
d_k &= -g_k + \beta_k d_{k-1}, \\
\beta_k &= \begin{cases} 0, & \text{for } k = 0, n, 2n, ..., \\ \frac{\|g_k\|^2}{\|g_{k-1}\|^2}, & \text{for } k \neq 0, n, 2n, ..., \end{cases}
\end{aligned}$$

with

$$g_k = \nabla f(x_k).$$

This conjugate gradient method with restart possesses has the following global convergence property [94]:

Theorem 5.7 *Let $f(x)$ be continuously differentiable and the set $\{x : f(x) \leq f(x_0)\}$ be bounded. Then, the method (5.29) has the global convergence property, that is,*

$$\nabla f(x_k) \to 0, \quad as \quad k \to \infty.$$

In the following Theorem, it turns out that this method converges with the quadratic rate in a neighborhood of the minimizer of $f(x)$.

Theorem 5.8 *Let x^* be a non-singular minimizer of $f(x)$ and let $\nabla^2 f(x)$ satisfy a Lipschitz condition in a condition in a neighborhood of x^*. Then, for the method (5.29), in a neighborhood of x^*, one has the estimate*

$$\|x^{(m+1)n} - x^*\| \leq c\|x^{mn} - x^*\|^2, \tag{5.30}$$

for some constant c.

From the above Theorem, in other words, with respect to the rate of convergence, the n steps of the conjugate gradient method are equivalent to one step of Newton's method.

There are more computational schemes for the conjugate gradient method for the non-quadratic problems. We now discussed one of these schemes, which requires the solution of a two-dimensional minimization problem at each step. The other schemes, which are similar to the method (5.28), usually include only one-dimensional auxiliary problems, but they differ from (5.28), in the rule for choosing the values of β_k. The following scheme is an example for this [23]:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{5.31}$$

where

$$\begin{aligned} \alpha_k &= \min_{\alpha \geq 0} f(x_k + \alpha d_k), \\ d_k &= -g_k + \beta_k d_{k-1}, \\ \beta_k &= \frac{(g_k - g_{k-1})^T g_k}{\|g_{k-1}\|^2}, \quad \text{for } k \geq 1, \\ \text{and} \quad \beta_0 &= 0. \end{aligned}$$

with

$$g_k = \nabla f(x_k).$$

The above computational scheme was proposed in [91] and [93]. Similar to (5.28), a variant with restart or without restart is also possible. For the quadratic function, the sequences x_k generated by the methods (5.28) and (5.31) are also the same.

Moreover, some scientists show that by the numerical computations, for the non-quadratic problems, the method in (5.31) usually gives a slightly faster con-

vergence.

Finally, we refer to [65] for the detailed investigation of the properties of the conjugate gradient method and of modifications of it, as well as a comparison of varied computational schemes.

5.3.2 Convergence Analysis of Conjugate Gradient Method

Here, following [26], we discuss the convergence with the rate of geometric progression for the case for the quadratic problem. Let A be an $n \times n$ matrix which satisfies

$$lI \leq A \leq LI, \tag{5.32}$$

where $0 < l < L$ are constants.
Let $f(x)$ be the corresponding quadratic function on $\mathbb{R}^n$, that is,

$$f(x) = \frac{1}{2}x^T Ax - b^T x + c, \tag{5.33}$$

for $b \in \mathbb{R}^n$ and c is a scalar.

Then, in the kth iteration, x_k can be represented in the following form:

$$x_k - x^* = P_k(A)(x_0 - x^*),$$

where $P_k(A)$ is a matrix polynomial of degree k of the form:

$$P_k(A) = I + a_{1k}A + ... + a_{kk}A^k.$$

Note that in the kth iteration of the conjugate gradient method, the iterative point x_k is the minimizer of $f(x)$ on the subspace which is passing through x_0

and generated by $g_0, ..., g_{k-1}$. Then, the polynomial $P_k(\lambda)$ satisfies the following condition:

$$\begin{aligned} 2(f(x_k) - f(x^*)) &= (x_0 - x^*)^T A P_k^2(A)(x_0)(x_0 - x^*) \\ &\leq (x_0 - x^*)^T A R^2(A)(x_0 - x^*), \end{aligned}$$

where $R(\lambda)$ is an arbitrary polynomial of degree k with the initial condition $R(0) = 1$.

Hence, we have:

$$\begin{aligned} \|x_k - x^*\|^2 &\leq \frac{2(f(x_k) - f(x^*))}{l} \\ &\leq \frac{\|A\|\|R^2(A)\|\|x_0 - x^*\|^2}{l} \\ &\leq \frac{L}{l}\|x_0 - x^*\| \max_{l \leq \lambda \leq L} R^2(\lambda). \end{aligned} \tag{5.34}$$

For the choice for the polynomial $R(\lambda)$ of degree k, we wish to have the least deviation from 0 on $[l, L]$ and with the initial condition $R(0) = 1$. For such a polynomial, we have:

$$R(\lambda) = \frac{T_k\left(\frac{L+l-2\lambda}{L-l}\right)}{T_k\left(\frac{L+l}{L-l}\right)}, \tag{5.35}$$

where $T_k(x)$ is the Chebyshev polynomial

$$T_k(x) = \begin{cases} \frac{1}{2}\left[(x + \sqrt{x^2 - 1})^k + (x - \sqrt{x^2 - 1})^k\right], & \text{if} |x| > 1; \\ \cos(k \arccos x), & \text{if} |x| \leq 1. \end{cases} \tag{5.36}$$

Then, we have:

$$\begin{aligned} \max_{l \leq \lambda \leq L} R^2(\lambda) &= T_k^{-2}\left(\frac{L+l}{L-l}\right) \max_{|x| \leq 1} T_k^2(x) \\ &= T_k^{-2}\left(\frac{L+l}{L-l}\right) \\ &= 4(q^k + q^{-k})^{-2} \\ &\leq 4q^{2k}. \end{aligned}$$

where $q = \frac{\sqrt{L}+\sqrt{l}}{\sqrt{L}-\sqrt{l}}$.

Hence, we have:

$$\|x_k - x^*\| \leq 2\left(\frac{L}{l}\right)^{\frac{1}{2}} q^k \|x_0 - x^*\|. \tag{5.37}$$

From the above result, for $k < n$, the conjugate gradient method used to minimize a quadratic function, one can guarantee a convergence with the rate of geometric progression with ratio

$$q = \frac{\sqrt{L} - \sqrt{l}}{\sqrt{L} + \sqrt{l}} \approx 1 - 2\sqrt{\mu},$$

where $\mu = \frac{L}{l}$.

From the above result, we can see that the conjugate gradient method is the same as the heavy ball method for the choice of its parameters. Versus the latter method, in the conjugate gradient method, the choice of parameters presents no problem. They are determined automatically, although they involve additional computations for solving the one-dimensional minimization problem.

It is obvious that in the conjugate gradient method, the vector x_k is a minimizer of the quadratic function $f(x)$ on the subspace generated by the first k gradients. It then follows that no method using only gradients of the function can converge more rapidly. In other words, the conjugate gradient method is optimal with respect to its rate of convergence in the class of the first order methods. The result obtained above implies that for the large scale problems with the quadratic functions $f(x)$ satisfying the condition (5.32), for all first order methods one cannot expect convergence of a higher rate than the rate of geometric progression with ratio $q = \frac{\sqrt{L}-\sqrt{l}}{\sqrt{L}+\sqrt{l}}$. Naturally, a higher rate of convergence can neither be attained in the broader class of strongly convex functions with constant l, which the gradient satisfies a Lipschitz condition with the Lipschitz constant L. In Theorem 5.8, the quadratic convergence occurs only when the number of iterations

is significantly greater than the dimension of the space.

Finally, we refer to [22] and [25] for the detailed investigation of the general convergence estimate for the conjugate gradient method. For more discussions on pre-conditioners and their construction, we refer to [3], [8], [60] and [74].

5.4 Methods of Variable Metric and Methods of Conjugate Directions

Let us see how the choice of the metric affects the form and the properties of the gradient method ([103], [55] and [27]) . Suppose that in the space $\mathbb{R}^n$ in addition to the initial product (x, y) a scalar product defined by a matrix A, which is positive definite, is given by:

$$(x, y)_1 = y^T A x. \tag{5.38}$$

In this case, the matrix A defines a new metric in $\mathbb{R}^n$:

$$\|x - y\|_1^2 = (x - y)^T A (x - y). \tag{5.39}$$

Let us write the gradient of a differentiable function $f(x)$ in the new metric:

$$\begin{aligned} f(x + y) &= f(x) + y^T \nabla f(x) + o(\|y\|) \\ &= f(x) + y^T A A^{-1} \nabla f(x) + o(\|y\|) \\ &= f(x) + (a, y)_1 + o(\|y\|_1) \end{aligned}$$

where

$$a = A^{-1} \nabla f(x).$$

By definition, the vector a is the gradient of $f(x)$ in space with scalar product (5.38). Therefore,

$$\nabla_1 f(x) = A^{-1}\nabla f(x). \tag{5.40}$$

In the new metric, the gradient method assumes the form

$$\begin{aligned} x_{k+1} &= x_k - \alpha_k \nabla_1 f(x) \\ &= x_k - \alpha_k A^{-1}\nabla f(x_k) \end{aligned} \tag{5.41}$$

and differs from the original gradient method by the presence of the matrix A^{-1}. In order words, the gradient method is not invariant with respect to the choice of matric of the space. It is reasonable to choose the metric such as to increase the rate of convergence. For the quadratic function

$$\begin{aligned} f(x) &= \frac{1}{2}x^T Bx - x^T b \\ &= \frac{1}{2}(A^{-1}Bx, x)_1 - (A^{-1}b, x)_1, \end{aligned} \tag{5.42}$$

the convergence rate of (5.41) is determined by the progression ratio

$$q = (L - l)(L + l),$$

where L and l are the largest and the smallest eigenvalues of $A^{-1}B$ respectively. If the matrix $A^{-1}B$ is closer to the unit matrix I, then the value of q becomes smaller. The best way is to choose $A = B$, since $A^{-1}B = I$, then $q = 0$, that is, if one defines the matric with the matrix B, then the gradient method will yield an accurate solution in one step with $\alpha_k \equiv 1$. This is not surprising, for $f(x) = \frac{1}{2}(x, x)_1 - (A^{-1}b, x)_1$, that is, the level lines of the function $f(x)$ are spheres and the condition number μ is equal to one.

For a non-quadratic function, follows from [49], the method

$$x_{k+1} = x_k - \alpha_k H_k \nabla f(x_k), \tag{5.43}$$

where H_k is positive definite, can be viewed as the gradient method in the following metric:

$$(x, y)_1 = (H_k^{-1}, y), \tag{5.44}$$

and $H_k = [\nabla^2 f(x_k)]^{-1}$ is the optimal choice of the metric. In other words, the Quasi-Newton methods (see Chapter 6) can be treated as gradient methods in which a new metric is chosen on each step as close to the best one as possible. For this reason, the term "methods of a variable metric" is often synonymous to that of quasi-Newton methods.

This interpretation is also useful as a heuristic construction of new variants of quasi-Newton methods. For example, one can obtain a new metric by extending the space in the direction of the last gradient, or in the direction of the difference of two consecutive gradients, and the like.

The another approach to constructing efficient first-order methods involves the notion of conjugate directions. As was observed in Section 5.3, the knowledge of the set of conjugate directions $d_0, ..., d_{n-1}$:

$$d_j^T A d_i = 0, \quad \text{for } i \neq j, \tag{5.45}$$

makes it possible to find the minimum of a quadratic function $f(x) = \frac{1}{2}x^T Ax - x^T b$ in n one-dimensional minimizations:

$$x_{k+1} = x_k + \alpha_k d_k \tag{5.46}$$

where

$$\alpha_k = \min_{\alpha} f(x_k + \alpha d_k).$$

Then, $x_n = x^* = A^{-1}b$, for any initial condition $x_0 \in \mathbb{R}^n$. One of the methods for constructing conjugate directions was used in the conjugate gradient method [49]:

the sequentially computed gradients were subjected to the A-orthogonalization. Other methods are quite possible as well.

Let $d_1, ..., d_k, k < n-1$ be conjugate vectors that have been constructed,

$$d_j^T A d_i = 0, \quad \text{for } i \neq j, \tag{5.47}$$

for $i \geq 0$ and $j \leq k$. Let x_k be the corresponding points in the method (5.46). The next vector d_{k+1} must satisfy the relation:

$$(Ad_i)^T d_{k+1} = 0, \quad \text{for } i = 0, ..., k.$$

Since

$$d_i = \frac{x_{i+1} - x_i}{\alpha_i},$$

and

$$\begin{aligned} Ad_i &= \frac{\nabla f(x_{i+1}) - \nabla f(x_i)}{\alpha_i} \\ &= \frac{y_i}{\alpha_i}, \end{aligned}$$

this is equivalent to the condition

$$y_i^T d_{k+1} = 0, \quad \text{for } i = 1, ..., k. \tag{5.48}$$

Therefore, the new conjugate direction d_{k+1} must satisfy the orthogonality conditions (5.48). Orthogonalization of any linearly independent vectors gives us varied sets of conjugate directions.

Similarly, for the non-quadratic function, we have the same process as follows:

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k d_k, \\ \alpha_k &= \min_{\alpha \geq 0} f(x_k + \alpha d_k), \\ y_i^T d_{k+1} &= 0, \quad \text{for } i = 1, ..., k, \\ y_i &= \nabla f(x_{i+1}) - \nabla f(x_i). \end{aligned} \tag{5.49}$$

Usually, d_{k+1} is sought here in the form

$$\begin{aligned} d_{k+1} &= -H_{k+1}\nabla f(x_{k+1}), \\ H_{k+1} &= H_k + \Delta H_k, \end{aligned} \tag{5.50}$$

and the matrix H_k is stored instead of the vectors y_i, $i = 1, ..., k$. The methods thus assume the same form as the quasi-Newton methods as follows (see Chapter 6):

$$x_{k+1} = x_k - \alpha_k H_k \nabla f(x_k).$$

The only difference is that it is not necessarily $H_k \to [\nabla^2 f(x_k)]^{-1}$; in some variants of the method, for example, the quadratic function,we have $H_n = 0$. That is why in these methods one must use a restart procedure.

We will next write an algorithm for one of the simplest methods of this class [49]:

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k d_k, \\ \alpha_k &= \min_{\alpha \geq 0} f(x_k + \alpha d_k), \\ d_k &= -H_k \nabla f(x_k), \\ y_k &= \nabla f(x_{k+1}) - \nabla f(x_k), \\ H_{k+1} &= H_k - \frac{H_k y_k (y_k)^T H_k}{(y_k)^T H_k y_k}, \quad \text{for} \quad k+1 \neq n, 2n, ..., \\ H_0 &= H_n = H_{2n} = ... = I. \end{aligned} \tag{5.51}$$

It turns out that for a quadratic function in method (5.51) the vectors d_k are conjugate directions, B_k is positive or semi-positive definite for all $k \leq n$, and $B_n = 0$. For the non-quadratic functions, the local convergence of methods of this class in a neighborhood of a nonsingular minimum point has been proved.

5.5 Other Approaches for Constructing the First-order Methods

Regardless the variety of first-order algorithms the idea behind them was the same for all of them, that is, to use a quadratic approximation of the function near the minimum. As a rule, these algorithms are finite for quadratic functions and in the general case, they are more efficient if their function is closer to being quadratic. But the quadratic model can be regarded to be natural only in a neighborhood of the extremum; far from the extremum the behavior of the objective function may be somewhat different. Hence, for all of the methods, it is clearly not advisable to apply an optimization strategy even at the initial stages of the search.

Instead, it is advantageous to use models of functions other than quadratic. It seems natural to make an attempt to construct polynomial models using higher derivatives: the next terms of the Taylor series. This has been tried before - however without good results. First, a direct computation of higher derivatives in multi-dimensional problems is usually too cumbersome and requires large memory. Furthermore, to reconstruct them from lower derivatives one needs to compute them at too large number of points. Secondly, auxiliary problems of minimizing polynomial functions cannot, with rare exception, be solved in the analytic form.

A simple and important class of models includes those based on the approximation of a homogeneous function [72]. The function $f(x)$, $x \in \mathbb{R}^n$, is called homogeneous with respect to x^* with the exponential $\gamma > 0$ if

$$f(x^* + \lambda(x - x^*)) - f(x^*) = \lambda^\gamma (f(x) - f(x^*)) \tag{5.52}$$

for all $x \in \mathbb{R}^n$ and $\lambda \geq 0$.

From (5.52), if we take $\lambda = 1 + \varepsilon$, where $\varepsilon \to 0$, then we have:

$$\begin{aligned} f(x + \varepsilon(x - x^*)) - f(x^*) &= (1 + \varepsilon)^\gamma (f(x) - f(x^*)), \\ \varepsilon\gamma(f(x) - f(x^*)) &= \varepsilon(x - x^*)^T \nabla f(x) + o(\varepsilon). \end{aligned}$$

Let $\varepsilon \to 0$, then this yields an important relation for the differentiable homogeneous function as follows:

$$f(x) - f(x^*) = \frac{(x - x^*)^T \nabla f(x)}{\gamma}. \tag{5.53}$$

The point x^* is not necessarily a minimum point of $f(x)$. However, if $f(x)$ attains a minimum, then x^* is a global minimum point of $f(x)$. Indeed, let $f(\tilde{x}) = f^* = \min f(x)$. Then, $\nabla f(\tilde{x}) = 0$. Substituting $\tilde{x}$ for x into (5.53), we get $f(x^*) = f(\tilde{x}) = f^*$, that is, x^* is a global minimum point.

Using (5.53), one can find the minimum point x^* through computation of $f(x)$ and $\nabla f(x)$ at a finite number of points. Indeed, if γ is known, then taking $n + 1$ points $x_0, ..., x_n$, yields the following system:

$$\gamma f(x_i) - \alpha + (x^*)^T \nabla f(x_i) = x_i^T \nabla f(x_i), \tag{5.54}$$

for $i = 0, ..., n$, which is linear in the $n + 1$ variables x^* and α, where $\alpha = \gamma f(x^*)$. By eliminating α, we obtain n linear equations to determine $x^* \in \mathbb{R}^n$:

$$(x^*)^T(\nabla f(x_i) - \nabla f(x_0)) = x_i^T \nabla f(x_i) - x_0^T \nabla f(x_0) - \gamma(f(x_i) - f(x_0)), \tag{5.55}$$

for $i = 1, ..., n$. But if γ is known, then one can take $n + 2$ points $x_0, .., x_{n+1}$, and determine the $n + 1$ variables γ, x^* from the linear system (5.55) in which $n + 1$ equations have to be taken.

A similar approach can be applied to minimize functions of general form, as in the secant method (in next chapter). Indeed, let the approximations $x_0, ..., x_k$,

where $k > n$, have been constructed. Taking the last $n+1$ among them (or $n+2$ if γ is known), we solve the following system with respect to x, α, γ or x, α:

$$x^T \nabla f(x_i) - \alpha + \gamma f(x_i) = x_i^T \nabla f(x_i), \tag{5.56}$$

where $i = k, k-1, ...,$ and for x_{k+1}, we take the solution x. For $\gamma = 2$, we get a method similar to the secant method, but not exactly the same - unlike the secant method, the method obtained uses both $\nabla f(x_i)$ and the values of the function $f(x_i)$.

Such a process should be modified using the same techniques as for the secant method, for example, eliminating the degeneration of points x_k by adding new points which are linearly independent of the preceding points; or adjusting the step size. A comparison of the actual value of $f(x_{k+1})$ with the predicted value (equal to $\frac{\alpha}{\gamma}$) is also useful to verify the assumption concerning the proximity of the function to being homogeneous. In solving systems of linear equations in successive iterations.

To minimize homogeneous functions or functions close to being homogeneous, some other methods can be used, for example, in the gradient method one uses special techniques for choosing the step size. Let the function $f(x)$ satisfy the condition (5.53), with the $f* = f(x^*)$ and γ being known. We consider the following gradient method:

$$x_{k+1} = x_k - \frac{\gamma(f(x_k) - f^*)}{\|\nabla f(x_k)\|^2} \nabla f(x_k). \tag{5.57}$$

The step

$$\alpha_k = \frac{\gamma(f(x_k) - f^*)}{\|\nabla f(x_k)\|^2}$$

is chosen such that the equality $f(x_k) - f^* = \frac{(x_k - x_{k+1})^T \nabla f(x_k)}{\gamma}$ is satisfied for

$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$. Then,

$$\begin{aligned} \|x_{k+1} - x^*\|^2 &= \|x_k - x^*\|^2 - \frac{2\gamma(f(x_k) - f^*)}{\|\nabla f(x_k)\|^2}(x_k - x^*)^T \nabla f(x_k) + \frac{\gamma^2 (f(x_k) - f^*)^2}{\|\nabla f(x_k)\|^2} \\ &= \|x_k - x^*\|^2 - \frac{\gamma^2 (f(x_k) - f^*)^2}{\|\nabla f(x_k)\|^2} \end{aligned}$$

implying that if $\|\nabla f(x)\|$ is bounded on the set $\{x : \|x - x^*\| \leq \|x_0 - x^*\|\}$, then $f(x_k) \to f^*$. It is not hard to see that this result still holds if in (5.53) equality is replaced by the following inequality:

$$f(x) - f^* \leq \frac{(x - x^*)^T \nabla f(x)}{\gamma}. \tag{5.58}$$

A somewhat different class, versus the homogeneous one, is given by the formula:

$$\begin{aligned} f(x) &= F(\phi(x)), \\ \text{where} \quad \phi(x) &= \frac{1}{2} x^T A x - x^T b, \end{aligned} \tag{5.59}$$

with A is positive definite and $F : \mathbb{R} \to \mathbb{R}$ is a monotone function on $[\phi^*, \infty)$, where $\phi^* = \phi(x^*)$. Obviously, x^* is a minimum point of $f(x)$.

If F and ϕ are given in the explicit form, a simpler problem of minimizing $\phi(x)$ can be solved instead of the problem of minimizing $f(x)$. In general, however, the information on the problem is not sufficient. Then, the following variant of the conjugate-gradient method can be used [72]:

$$\begin{aligned} x_{k+1} &= x_k + \alpha_k d_k \\ \alpha_k &= \min_{\alpha \geq 0} f(x_k + \alpha d_k), \\ d_k &= -\nabla f(x_k) + \beta_k d_{k-1}, \\ \beta_k &= \frac{F'(\phi(x_{k-1}))\|\nabla f(x_k)\|^2}{F'(\phi(x_k))\|\nabla f(x_{k-1})\|^2}, \\ \beta_0 &= 0. \end{aligned} \tag{5.60}$$

Note that the method (5.60) generates the same sequence of points as the conjugate-gradient method does for minimization of $\phi(x)$; it is therefore finite.

The quantity

$$\rho_k = \frac{F'(\phi(x_k))}{F'(\phi(x_{k-1}))}$$

in the formula for β_k can be estimated approximately via approximation of the $F(z)$ by a quadratic or a power function. In that case, the method (5.60) can be used to minimize functions that do not necessarily have the form (5.59).

Finally, we conclude this section by the following convergence property related to the above algoithm [72]:

Theorem 5.9 *Let $f(x)$ be a twice continuously differentiable function defined on $\mathbb{R}^n$ and let $f(x) \to \infty$ as $\|x\| \to \infty$. Furthermore, let $f(x)$ have a unique stationary point x^* which is also its unique minimizer. Then, the algorithm (5.60) generates a monotone decreasing sequence $\{f(x_k)\}$. Also, the sequence $\{x_k\}$ is bounded.*

Chapter 6

Quasi-Newton Methods

We begin by recalling the simplest form of Newton's method in Chapter 4. The minimization problem is as follows:

$$\min_{x \in \mathbb{R}^n} f(x), \tag{6.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a differentiable function.

When we use Newton's method, the iterative formula in the kth iteration becomes:

$$x_{k+1} = x_k - \alpha_k \left(G_k\right)^{-1} g_k, \tag{6.2}$$

where x_k is the kth approximation to the solution, g_k and G_k denote the first and second derivatives of $f(x)$ evaluated at x_k respectively, and α_k is the steplength of the kth iteration.

If $f(x)$ is known to be convex, then any stationary value of $f(x)$ is a minimum, and G_k is either positive definite or semi-definite. Therefore, it can provide the values of the first and second derivatives of $f(x)$, which can be computed for any

given $x \in \mathbb{R}^n$, and it never becomes singular. So, Newton's method may be used to determine a minimum. The advantages of the Newton's method are [86]:

(1) If Newton's method works at all, then it works extremely well;

(2) The convergence is rapid and in general is ultimately quadratic;

(3) If a sufficiently good initial estimate of the solution can be determined, it is probably the best available method.

6.1 Disadvantages of Newton's Method

Although Newton's method has many advantages in computation, it is not without its disadvantages and now we consider three of these in particular.

Refer to [86], perhaps the most serious charge levelled against this form of the method is that it often fails to converge to a solution from a poor initial estimate. The successive iterates change in an apparently quite random fashion and not infrequently a value of x_k is computed that leads to some kind of failure, for example, the taking of a square root of a negative number, when evaluating $\nabla f(x)$. To overcome this problem we normally choose the parameter α_k so that in some sense x_{k+1} may be said to be a better approximation to the solution of the problem than x_k. If, for instance, we are concerned with minimizing $f(x)$, we may choose α_k so that $f(x_{k+1}) < f(x_k)$. A choice of this nature requires for its implementation that some form of iteration or search procedure be carried out. It follows from the formula (6.2) that this search is along a straight line in n-dimensional Euclidean space and hence this part of any optimization procedure is known as the "line search". If it is necessary to choose α_k to minimize $f(x)$ as a function of α, then we can say that an "exact line search" is required. However, the details choice of this scaling factor does not affect the basic philosophy behind

the quasi-Newton methods and we will discuss it no further at this stage. We return to it later in connection with some particular algorithms.

The second disadvantage of Newton's method is the difficulty of evaluating $G(x)$ if $\nabla f(x)$ is a complicated function of x. In many industrial problems, it is virtually impossible to obtain the elements of $G(x)$ as explicit expressions and even if it were possible it would, for some problems, be an extremely laborious and time-consuming operation. In these circumstances, we are condemned to using some approximation B to $G(x)$, and the formula (6.2) then becomes:

$$x_{k+1} = x_k - \alpha_k [B(x_k)]^{-1} \nabla f(x_k). \tag{6.3}$$

One method of obtaining $B(x_k)$ is by the use of forward differences, computed its ith column using the equation:

$$B(x_k)e_i = \frac{g(x_k + he_i) - g(x_k)}{h}, \tag{6.4}$$

where $g(x) = \nabla f(x)$, and e_i is the ith column of the unit matrix of order n and h is a suitably small scalar. However, this is an expensive use of machine time since in order to evaluate $G(x)$ in this somewhat simple-minded fashion, it is necessary to compute the vector function $\nabla f(x)$, $n + 1$ times. Since one value of $\nabla f(x)$ needs to be calculated anyway (in order to compute x_{k+1} using the formula (6.3)), this implies n additional evaluations of $\nabla f(x)$.

A sophisticated approach has been adopted by Brown and Conte [13], which reduces the amount of additional labour by about half, but even in this case there is a very considerable penalty at each iteration. A further weakness of the forward difference approach is the essentially empirical choice of h. In the absence of rounding error h should be reduced to zero as the solution is approached [107], but when rounding error is present, as it invariably is, h should in principle be chosen to minimize the sum of rounding and truncation errors. In practice, it is

usually chosen arbitrarily in the range 10^{-3} to 10^{-6}. Perhaps we should emphasize here though that these disadvantages are not sufficient to condemn methods of this type out of hand. It may well be possible that the total number of iterations using them is sufficiently small to make them competitive, and the evidence suggests that the more difficult the problem is, the more likely this reduction in the number of iterations becomes.

Another way of overcoming the difficulty of computing $G(x)$ is to evaluate an approximation to it numerically only at every m iterations, where m is some positive integer. The problem here lies in knowing how to choose m, and even if the best choice of m is made the method may still be inferior to methods of the quasi-Newton type. For these reasons, this idea is hardly ever implemented today.

The third disadvantage of Newton's method is the necessity of solving a set of linear system equations at each iteration. This is perhaps the least of the three disadvantages that we have recorded, but it does take both extra time and extra programming, although these undesirable features are perhaps not so important now that we have very fast computers with large core stores.

6.2 General Idea of Quasi-Newton Method

6.2.1 Quasi-Newton Methods

We now look rather more closely at another way of overcoming the second disadvantage of Newton's method, and present the idea which provides the underlying motivation for the Quasi-Newton methods. Follow [128], we let, B be some approximation to $G(x)$ and let us compute x_{k+1} using the formula (6.3). We now

consider methods which enable us to obtain better approximations to $G(x)$ without any additional function evaluations. To expose the underlying idea behind these methods, we consider $f(x)$ to be the function minimization defined by

$$f(x) = \frac{1}{2}x^T Ax - b^T x + c, \tag{6.5}$$

where A is a constant matrix which is symmetric and positive definite, b is a constant vector and c is a scalar. Since in this case, $\nabla f(x) = Ax - b$, we can identify $\nabla f(x)$ and the equation $\nabla f(x) = 0$, which requires the gradient to be the null vector, becomes the condition for a stationary value of $f(x)$.

Now, we consider the equation

$$\nabla f(x) = Ax - b = 0. \tag{6.6}$$

It is clear that the second derivative of $f(x)$ defined in this way is simply A, so that if s_k and y_k are defined by

$$s_k = x_{k+1} - x_k, \tag{6.7}$$

$$y_k = \nabla f(x_{k+1}) - \nabla f(x_k), \tag{6.8}$$

then $G(x)$ satisfies the equation

$$G(x_k)s_k = y_k. \tag{6.9}$$

Since $B(x_k)$ is some approximation to G_k, we would like it also to satisfy equation (6.9), but since we cannot compute $\nabla f(x_{k+1})$ and hence y_k, until we have determined $B(x_k)$, this is clearly impossible. We can require that the next approximation to $G(x)$, namely $B(x_{k+1})$, satisfies

$$B(x_{k+1})s_k = y_k, \tag{6.10}$$

so that $B(x_{k+1})$ has at least one property of $G(x_k)$. In the case where the functions $\nabla f(x)$ are non-linear so that $G(x)$ is not constant, and the equation (6.9)

ceases to be valid, the same sort of considerations apply. In the equation (6.9), $G(x_k)$ may be replaced by $\bar{G}(x_k)$, where $\bar{G}(x_k)$ is a matrix whose ith row is the ith row of $G(x_k + s_k\theta_i)$, for some θ_i satisfying $0 < \theta_i < 1$, so that $\bar{G}(x_k)$ approximates $G(x_k)$ to an accuracy which depends both on $\|s_k\|$ and the non-linearity of $f(x)$. The quasi-Newton equation in this case forces $B(x_{k+1})$ to assume one property of $\bar{G}(x_k)$. Since for non-linear systems $G(x)$ and $\bar{G}(x)$ are not constant, this ensures that the approximation B reflects changes in $G(x)$ and $\bar{G}(x)$ and the hope is that this will assist in obtaining rapid convergence. The equation (6.10), the quasi-Newton equation, is the equation underlying all the algorithms discussed later. Since it does not define B_{k+1} uniquely, it applies to a class of methods where the properties of the individual methods vary with the particular choice of $B(x_{k+1})$. The methods that we discuss have common properties other than that of satisfying the quasi-Newton equation, and we now turn our attention to these.

Clearly, if $B(x_k)$ is a reasonably good approximation to $G(x_k)$, then it is advantageous that $B(x_{k+1})$ should retain as far as possible the desirable properties of $B(x_k)$. This suggests that $B(x_{k+1})$ is formed from $B(x_k)$ by adding a correction term C_k so that

$$B(x_{k+1}) = B(x_k) + C_k, \tag{6.11}$$

and that some criteria, for example, the requirement that the quasi-Newton equation be satisfied, be selected in order to more precisely determine C_k. The criteria that we consider here for specifying C_k, are:

(1) C_k to be a rank-1 matrix;

(2) C_k to be a rank-2 matrix;

(3) C_k to be a matrix of minimum norm.

The single-rank corrections have been considered by, among others, Broyden [15], Davidon [28], Murtagh and Sargent [88] and Pearson [90], double-rank corrections

by Broyden [17], Davidon [29], Flectcher and Powell [50], Fletcher [46], Powell [99], and Pearson [90], and minimum-norm corrections by Goldfard [58], and Greenstadt [63]. Since the majority of minimum-norm corrections are in fact either rank-1 or rank-2 corrections, we will not consider them specifically from the minimum-norm point of view. We do remark that the fact that the same algorithm may be derived by applying apparently unrelated criteria does inspire a certain amount of confidence in its use.

We have chosen here to explain the quasi-Newton methods by relating them to Newton's method, and this is representative of their historical development. But it is readily seen that any sequence of vectors x_k, no matter how it is generated, and the associated sequence of functions $\nabla f(x_k)$ may be used, from the equations (6.7) to (6.11), to construct approximations to the second derivative of $f(x)$. Moreover, this may be done in the case where ∇f is an m-vector, x is an n-vector, where $m \neq n$, that is in the case where the second derivative of f is rectangular. This case is of practical importance when solving the over-determined least-squares problem, where $m > n$, or in minimizing a function subject to equality constraints, where $m < n$. In both these cases matrix updates more commonly associated with Newton's method have been used to obtain estimates of the second derivative of f, and examples of this usage are considered later.

This approach has been particularly successful for least squares problems. Here, the basic iteration is the Gauss-Newton one, where:

$$x_{k+1} = x_k - \left(G(x_k)^T G(x_k)\right)^{-1} G(x_k)^T \nabla f(x_k). \tag{6.12}$$

The second derivative has more rows than columns, that is, $m > n$, but even so approximately satisfies the equation (6.9). If the second derivative is either difficult or expensive to evaluate, then as before it may be replaced in the equa-

tion (6.12) by an approximation $B(x_k)$; and as before $B(x_{k+1})$ may be made to satisfy the quasi-Newton equation.

We have not yet suggested how the disadvantages of solving a set of linear system of equations at each step may be overcome and for some problems, particularly over-determined least-squares problems, no way has yet been found of avoiding this. In the case where the second derivative of f, that is $G(x)$, is square and non-singular, as well as where Newton's iteration is used a technique has been devised to replace the solving of equations by a matrix-vector multiplication. This technique is based upon the observation that if a matrix is modified by adding a correction of rank r, then its inverse may also be modified by adding a correction of rank r. Therefore, instead of storing and modifying an approximation $B(x_k)$ to $G(x)$, it is sufficient to store and modify an approximation to the inverse of $G(x)$. If this is denoted by H_k, then the equation (6.3) becomes:

$$x_{k+1} = x_k - \alpha_k H_k \nabla f(x_k), \tag{6.13}$$

and we have the most common form of the iteration. Despite the successes achieved by algorithms of this kind over the lest decade, development is still proceeding. If H_k is symmetric and positive definite so that $H_k = LL^T$ for some lower triangular matrix L, Gill and Murray [53] have suggested storing and modifying the triangular factor L. This approach seems to give the most useful improvements if $\nabla^2 f(x)$ is itself obtain using finite differences.

Algorithm 6.1 (General Qusai-Newton Methods)

Step 1: Given the initial conditions $x_0 \in \mathbb{R}^n$, $B(x_0) \in \mathbb{R}^{n \times n}$ *and* $k := 0$.

Step 2: Calculate $g_k = \nabla f(x_k)$.

Step 3: If $\|g_k\| \leq \varepsilon$, *then stop; otherwise, solve the equation* $B(x_k)d = -g_k$

to have the direction d_k.

Step 4: Along the direction d_k, *find the steplength* $\alpha_k > 0$ *such that*

$$x_{k+1} = x_k + \alpha_k d_k.$$

Step 5: Rectify $B(x_k)$ *to have* $B(x_{k+1})$, *which is satisfied* $B(x_{k+1})s_k = y_k$,

where $s_k = x_{k+1} - x_k$ *and* $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$.

Step 6: Set $k := k + 1$, *then go back to Step 2.*

6.2.2 Convergence of Quasi-Newton Methods

Now, we know that Quasi-Newton methods are based on the idea of reconstructing a quadratic approximation of a function from values of its gradients at a number of points. Those methods thereby combine the merits of the gradient method, that is, there is no calculation of the matrix of second derivatives or Hessian matrix, and those of Newton's method having rapid convergence as a result of quadratic approximation.

Let us note some general convergence properties of Quasi-Newton methods [94].

Lemma 6.1 *Let* $f(x) \geq f(x^*)$, *let* $f(x)$ *be differentiable, let* $\nabla f(x)$ *satisfy a Lipschitz condition and let*

$$mI \leq H_k \leq MI, \quad m > 0. \tag{6.14}$$

Then, in the method (6.13) with $\alpha_k \equiv \alpha$, *where* $\alpha > 0$ *is sufficiently small, one has* $\nabla f(x_k) \to 0$.

Lemma 6.2 *Let* x^* *be a nonsingular minimizer of* $f(x)$, *let* $f(x)$ *be twice con-*

tinuously differentiable in a neighborhood of x^* and let

$$\|H_k - [\nabla^2 f(x^*)]^{-1}\| \to 0. \tag{6.15}$$

Then, the method (6.13) with $\alpha_k \equiv 1$ converges locally to x^* faster than any geometric progression.

Therefore, for any uniformly positive definite H_k in the method (6.13) possesses global convergence, and under the condition (6.15), it converges in a neighborhood of the minimizer with super-linear rate.

Theorem 6.3 *For any x_0, H_0 is positive definite, the method (6.7), (6.8) and (6.13) with any of the update formula for H_k and*

$$\alpha_k = \min_{\alpha} f(x_k + \alpha d_k)$$

for the quadratic function $f(x) = \frac{1}{2}x^T Ax - x^T b + c$ is finite. That is,

$$x_n = x^* = A^{-1}b.$$

Furthermore, one can show that regardless the differences between the updating formulae for H_k, the sequences x_k generated by each variant of the method coincide for a quadratic function $f(x)$.

For non-quadratic functions $f(x)$, the Quasi-Newton methods in the form given above are usable, but they are no longer finite. Therefore, for $k > n$, one can either continue the computation by the same formulae, or begin a restart procedure, that is, replacing H_k by H_0 every n iterations.

Currently a super-linear or quadratic rate of convergence has been proved for many variants of Quasi-Newton methods in a neighborhood of a nonsingular minimizer.

6.3 Properties of Quasi-Newton Methods

In this part, follow from [86], we will consider the desirable and undesirable features of minimization algorithms and attempt to relate them to specific properties possessed by individual quasi-Newton methods. We begin by discussing briefly these features that we would take into consideration when seeking an algorithm to solve a given problem.

Perhaps the most important requirement is that the algorithm should not converge to an incorrect solution. A particular example of this would be convergence to a saddle-point instead of a minimum, but since this defect is not peculiar to quasi-Newton methods we consider it no further. Another feature to be avoided at all costs is premature or false convergence, and here it is possible that certain properties of the quasi-Newton methods could give rise to failure. It sometimes occurs that the matrix H becomes singular or nearly so, and the resulting steplength, $\|s\|$, becomes in consequence negligible. If one terminated the iteration by testing $\|s\|$ alone, then one could have false convergence, but this may be prevented by testing also $\|\nabla f\|$, the norm of the gradient of f. The quantity of interest is the norm of the error, but this can only be estimated. Let x^* be the solution of the solution of the system of linear equations so that

$$\nabla f(x^*) = 0, \tag{6.16}$$

and make the assumption that in the neighbourhood of x^*, $\nabla f(x)$ is essentially linear, satisfying the equation (6.6). Define the vector error e by

$$e = x - x^*. \tag{6.17}$$

Then, the equations (6.6) and (6.16) yield

$$Ax^* - b = 0. \tag{6.18}$$

So, for a good approximation in the neighourhood of x^*, then

$$\nabla f(x) = Ae, \tag{6.19}$$

where $A = G(x^*)$. Assume that A is non-singular and define the matrix E by:

$$E = B - A, \tag{6.20}$$

where B is the current approximation to A. If H is defined to be B^{-1}, it follows from the equations (6.19) and (6.20) that

$$e = (I - HE)^{-1} H. \tag{6.21}$$

Hence, if $\|HE\| < 1$, then we have:

$$\|e\| \leq \frac{\|H\|\|\nabla f\|}{1 - \|HE\|}. \tag{6.22}$$

Therefore, since $\|H\|$ and $\|\nabla f\|$ can be evaluated, if we suspect that $\|HE\| \ll 1$, then we can estimate an upper bound for the norm $\|e\|$. In order to use this device with some measure of confidence, we would require that, once a good approximation to $G(x^*)^{-1}$ had been achieved, it would not be spoiled by subsequent iterations. We would hope at least that $\|E\|$ would not increase as the iteration proceeded and we would indeed prefer it to be reduced. Whether or not an algorithm possesses, this property of matrix error norm reduction when it is used to minimize the quadratic functions may be readily established theoretically. It is a property that influences both the convergence and stability properties of an algorithm and is then a critical property to consider when attempting any assessment of merit.

Another property that we require of algorithms is that they should not fail catastrophically when updating the matrix H. Some algorithms have this property at all stages of the process and some at no stage. Some algorithms hope to

avoid this failure since all divisors used in implementing the update are known in theory to be non-zero, and this type of failure hardly ever occurs. Even in those algorithms for which a zero division is theoretically possible, its occurrence is infrequent. A more usual cause of failure in the quasi-Newton algorithms is a tendency for the algorithm to get stuck at a certain stage of the process, when changes in the current approximation to the solution become negligibly small. This behavior has been observed with many such algorithms ([5], [90], [17]) and is nearly always associated with the occurrence of a singular H. To see why a singular value of H should cause this behavior, we consider a general matrix updating formula which includes the most commonly used formulae as special cases.

For every algorithm discussed later, since

$$s_k = -\alpha_k H_k \nabla f(x_k), \tag{6.23}$$

write H_{k+1} as

$$H_{k+1} = H_k M_k, \tag{6.24}$$

where M_k is a matrix specific to a particular update. Then, by induction, we have:

$$H_{k+r} = H_k M_k M_{k+1} ... M_{k+r-1}, \quad \text{for } r \geq 1,$$

so that since

$$\begin{aligned} s_{k+r} &= -\alpha_{k+r} H_{k+r} \nabla f(x_{k+r}), \\ &= -H_k v, \quad \text{for } r \geq 1, \end{aligned} \tag{6.25}$$

where

$$v = \alpha_{k+r} M_k M_{k+1} ... M_{k+r-1} \nabla f(x_k).$$

Suppose that H_k is singular, then for some vector q, we have:

$$q^T H_k = 0.$$

It follows immediately form the equation (6.25) that $q^T s_{k+r} = 0$, for $r \geq 1$, so that once a particular H_k becomes singular subsequent steps are orthogonal to some fixed vector and hence are restricted to lie in a subspace of $\mathbb{R}^n$. In general, the solution does not lie in this subspace. But if it lies in this subspace, it will be completely unattained subsequent to the occurrence of a singular H. Although this or similar behavior has been observed in all the algorithms considered here some algorithms are more prone to this failing than others. it has general been overcome in practice by resetting H to be the unit matrix after every $2n$ iterations [90]. Another possibility is the use of the normal update after taking a step s_k other than that given by the equation (6.13) [17].

If those algorithms exist that avoid all the pitfalls described above one might then think in terms of obtaining more rapid convergence. Since Newton's method converges rapidly near the solution one might require an algorithm to resemble Newton's method as closely as possible, and this might lead to the requirement that the matrix error norms $\|E_k\|$, where $E_k = B(x_k) - A$, defined by the equation (6.20), decrease in some way when minimizing a quadratic function. This property generalizes in the case of non-quadratic functions to the property of "bound deterioration" [32]. If we define E_k by

$$E_k = B(x_k) - G(x^*),$$

where x^* is the solution to which the algorithm is converging, then for algorithms possessing the property of bounded deterioration some norm of E_k increases after a number of steps by an amount not exceeding some constant times the sum of the steplengths. Then, if $\|E_0\|$ is sufficiently small and x_0 is sufficiently close to x^*, the amount of deterioration in the accuracy of the approximation to the second derivative of f during the course of iteration will be insufficient to prevent convergence. Therefore, if the property of bounded deterioration can be established for an algorithm when applied to a class of problems, the construction of

a local convergence proof is a formality.

Another property that might lead to rapid convergence overall is the property of minimizing a quadratic function in at most n steps. The desirability or otherwise of this property of the quadratic termination and the analogous properties for solving general non-linear systems and constrained optimization problems, for example, linear termination, quadratic/linear termination, has not been fully established, and a certain amount of uncertainty still remains. Fletcher writes [45]: "These examples, and many others, suggest that the property of quadratic/linear termination is a desirable attribute to be aimed at when designing methods for non-linear programming", but we have, on the other hand, "The property of quadratic termination, whose relevance for general functions has always been questionable" [46]. The present author believes that the property of quadratic/linear termination is important provided that it is not achieved at the expense of stability, although he acknowledges that the evidence for this belief is stronger in the case of algorithms designed to solve general non-linear simultaneous equations than it is in the case of algorithms for function minimization.

Another feature that affects the overall speed of an algorithm is the amount of work required during each iteration. If α_k is taken to be unity in equations (6.3) and (6.13), then the amount of work is minimal. On the other hand, if it is necessary to minimize $f(x)$, then the computing cost of each iteration will be quite high since an inner iteration is needed at every step of the main iteration in order to compute α_k. If this inner iteration has to be carried out at all, it should clearly be carried out as efficiently as possible. However, a necessary feature for an efficient line search is the requirement that the vector - $H_k \nabla f(x_k)$ always points in the "downhill" (or "uphill") direction so that it is known before the search commences whether a positive (or negative) value of α_k will give the

required minimum. This knowledge may then be used in initiating the search for the minimum and may well contribute in the subsequent choice of search strategy. We show now, therefore, that if H_k is positive definite and x_{k+1} is given by the equation (6.13) then, for the value of α_k sufficiently small and positive, we have

$$f(x_{k+1}) < f(x_k), \quad \text{for all } k.$$

Let p be an arbitrary vector and α a scalar. Then, since

$$f(x_k + \alpha p) = f(x_k) + \alpha p^T \nabla f(x_k) + O(\alpha^2), \tag{6.26}$$

if we assume that $|\alpha|$ is sufficiently small so that terms in α^2 may be ignored, and we can always find such an α provided that f is continuous in a neighbourhood of x_k. Then, on substituting $-H_k \nabla f(x_k)$ for p and appealing to the equations (6.13) and (6.26), we have:

$$f(x_{k+1}) - f(x_k) = -\alpha_k f(x_k)^T H_k f(x_k).$$

Therefore, if H_k is positive definite, a positive value of α_k will result in a reduction of $f(x)$ whenever it is small, so that, provided $f(x)$ is continuous, a minimum of $f(x)$ will occur for a positive value of α. If we choose $\alpha_k = 1$ for all k, there is no guarantee in general that the process defined by the equation (6.13) and the relevant updating formulae will converge at all so that the additional work involved in minimizing $f(x)$ may be regarding as a premium to be paid in order to improve reliability. Recent tendencies, however, in Fletcher [46], have been to choose a value of α_k that merely reduces $f(x)$ despite the fact that this results in sacrificing the property of quadratic termination which relies, for many algorithms, upon exact minimization.

This last strategy illustrates the compromises forced upon the numerical analyst. No single algorithm embraces the properties of stability, quadratic termination and non-iterative determination of α. A choice therefore has to be made

and the overall performance of the algorithms studied. In this way we hope to establish which properties give rise to stable, rapidly converging algorithms so that we can devise future algorithms specifically with the idea of possessing these particular properties.

6.4 Some Particular Algorithms for Quasi-Newton Methods

In this part, we will consider particular algorithms for minimizing functions in the light of the discussion of the previous three sections. We will be principally concerned with properties such as quadratic termination, bounded deterioration and stability and will endeavour to relate these properties to the observed performance of the algorithms under discussion. We deliberately refrain from a detailed discussion of algorithms for solving non-linear simultaneous equations and constrained optimization problems. In general, we are content to give such algorithms only a brief mention unless they have been used in the context of unconstrained optimization or unless they form the basis of more sophisticated minimization algorithms.

6.4.1 Single-Rank Algorithms

Algorithm 1: Broyden's Algorithm [15]

This algorithm was intended for the solution of general sets of non-linear equations, but it apparently performs remarkably well when used to minimize functions [67]. The reason for its inclusion in this survey though is since its relation-

ship to algorithms that have been developed specifically to solve minimization problems. The update of the approximate second derivative $B(x_k)$ is given by

$$B_{k+1} = B_k - (B_k s_k - y_k) \frac{s_k^T}{s_k^T s_k}, \tag{6.27}$$

where

$$\begin{aligned} s_k &= x_{k+1} - x_k, \quad \text{and} \\ y_k &= \nabla f(x_{k+1}) - \nabla f(x_k), \end{aligned}$$

which is given by (6.7) and (6.8). So, if B is square and non-singular, it follows from the Sherman-Morrison formula that

$$H_{k+1} = H_k - (H_k y_k - s_k) \frac{s_k^T H_k}{s_k^T H_k y_k}. \tag{6.28}$$

It may be readily verified from the equations (6.20) and (6.27) that

$$E_{k+1} = E_k \left(I - \frac{s_k s_k^T}{s_k^T s_k} \right), \tag{6.29}$$

so, for the quadratic functions, the error matrix E_k decreases monotonically. The method is one of bounded deterioration and may be shown to be locally convergent for $\alpha_k = 1$ [32]. The algorithm is stable provided that B_k approximates the second derivative G_k sufficiently well [18], but if the approximation is poor, then the algorithm is unstable and its performance suffers in consequence. It is necessary to provide this algorithm with a starting procedure to ensure that the initial approximation is tolerably accurate. The algorithm does not enjoy the property of quadratic termination even when used for function minimization with exact line searches and α_k is usually chosen to reduce $\|f\|$ [15], or is set to be equal to unity [18]. The approximation H is not symmetric and neither is $H + H^T$ necessarily positive definite so, if line searches are used, this fact must be recognized by the line-search routine.

In practice, if the algorithm is in the region of local convergence with $\alpha_k = 1$, then it performs very well, convergence being usually super-linear. Broyden [18] showed that if f and e are given by the equations (6.6) and (6.17), $\alpha_k = 1$, for all k and $\|E_0\| < 1$, then we have:

$$\|e_r\| \leq \left(\frac{c}{r^{\frac{1}{2}}}\right)^r \|e_0\|, \tag{6.30}$$

where

$$c = \frac{\|E_0\|}{1 - \|E_0\|_{L_2}},$$

for $\|\cdot\|$ and $\|\cdot\|_{L_2}$ denote the Euclidean norm and the L_2 norm respectively.

A similar rate of convergence is observed for non-linear problems near the solution.

Algorithm 2: The Secant Algorithm ([7], [123])

This is the oldest of all the quasi-Newton methods, and which is another method that is not restricted to the minimization of functions, but may be used to solve a general set of non-linear equations. The updating equation is:

$$B_{k+1} = B_k - (B_k s_k - y_k) \frac{q_k^T B_k}{q_k^T B_k s_k}, \tag{6.31}$$

where q_k is given by

$$q_k^T y_j = 0, \quad \text{for} \quad k - n + 1 \leq j \leq k - 1. \tag{6.32}$$

Hence, if B is square and non-singular,

$$H_{k+1} = H_k - (H_k y_k - s_k) \frac{q_k^T}{q_k^T y_k}.$$

Since q_k is only defined by the above equations if at least $n - 1$ steps have taken place a starting procedure is needed (For example, Barnes [7]).

The algorithm as described has the property of linear termination since it follows from the equations (6.31) and (6.32), after some manipulation, that

$$B_{k+1}s_j = y_j, \quad \text{for } k-n+1 \leq j \leq k. \tag{6.33}$$

Therefore, B_{k+1} is determined uniquely by the previous n steps so that for linear functions, the approximation B becomes equal to the exact matrix G after n steps. A further Newton step then gives the exact solution, and we have termination after $n+1$ steps.

The difficulty with the secant method lies in the probability, for non-linear problems, that n consecutive y's or n consecutive s's will become linearly dependent. This deprives it of the property of bounded deterioration and in practice makes it notoriously unstable. Attempts to reduce this instability are usually based on requiring the equation (6.33) to be satisfied not for $k-n+1 \leq j \leq k$, but for n values of j chosen with stability in mind. Instead of replacing the "oldest" s and y by the "newest" pair (as in the equation (6.33) requires), the "newest" pair of s and y replaces a pair chosen so that the replacement impairs stability by the smallest amount. Therefore, the strict linear termination property is sacrificed to improve stability. The secant algorithm, modified or unmodified one, may be used with $\alpha_k = 1$ or with $\alpha_k = 1$ or with α_k chosen to reduce $\|f\|$, but the method does not appear to be extensively used in either of these modes. As with the previous method its interest, in the present context, lies in the application of its basic philosophy to algorithms more specifically orientated towards optimization.

Algorithm 3: The McCormick-Pearson Algorithm [90]

This class of methods is a generalization of two methods given by Pearson [90], one of which he attributes to McCormick. It is a sub-class of the 3-parameter

family of Huang [69], for which

$$H_{k+1} = H_k + \rho_k s_k (q_1)_k^T - H_k y_k (q_2)_k^T,$$

where

$$(q_i)_k = c_i s_k + k_i H_k^T y_k, \quad i = 1, 2.$$

Here, the scalar ρ_k is arbitrary and the scalars c_i and k_i are also chosen arbitrarily, but subject to the requirement that $(q_i)_k^T y_k = 1$, for $i = 1, 2$. It follows immediately that $H_{k+1} y_k = s_k \rho_k$ so that if $\rho_k = 1$, then B_k must satisfy the equation (6.10). Huang showed that if x_{k+1} is given by

$$x_{k+1} = x_k - \alpha_k H_k^T \nabla f(x_k) \qquad (6.34)$$

for all k and if α_k is always chosen to minimize $f(x)$, then all algorithms belonging to the class have the property of quadratic termination for any choice of the arbitrary parameters. Moreover, the steps s_k satisfy the equation

$$s_k^T A s_j = 0, \quad j \neq k, \qquad (6.35)$$

where A is the matrix defining the quadratic form given by the equation (6.5), so that the methods generate conjugate directions and terminate, for the quadratic functions, after at most n and not $n + 1$ steps.

The update for the McCormick-Pearson algorithm is given by

$$H_{k+1} = H_k - (H_k y_k - s_k) \left[\frac{\gamma_k s_k^T}{s_k^T y_k} + \frac{(1 - \gamma_k) y_k H_k}{y_k^T H_k y_k} \right], \qquad (6.36)$$

where γ_k is an arbitrary parameter which is zero for Pearson's and unity for McCormick's algorithm. Pearson proved that the algorithms are stable when minimizing a quadratic function, but he gave no proof of stability for general functions. It is likely that both McCormick's and Pearson's versions are of bounded deterioration and are locally convergent. The two versions were tested by Pearson [90],

and their performance was comparable with other methods.

Algorithm 4: The Symmetric Algorithm [28]

This algorithm, which has been investigated by Davidon [28] and Murtagh and Sargent [88], is the maverick not since it refuses to be categorized, but because it fits into so many categories. It is a member of the McCormick-Pearson class, a degenerate member of the rank-2 single-parameter family and it has some of the features of the secant method.

It is the only single-rank method which preserves the symmetry of B_k, and from this and the fact that B_{k+1} must satisfy the equation (6.10), it is readily deduced that

$$B_{k+1} = B_k - \frac{u_k u_k^T}{u_k^T s_k} \tag{6.37}$$

and

$$H_{k+1} = H_k - \frac{v_k v_k^T}{v_k^T y_k}, \tag{6.38}$$

where

$$\begin{aligned} u_k &= B_k s_k - y_k, \quad \text{and} \\ v_k &= H_k y_k - s_k. \end{aligned}$$

The basic method is not of bounded deterioration and is notoriously unstable in its unmodified form. Davidon suggests various updating strategies and makes a choice after performing certain tests, and Murtagh and Sargent use a different reset strategy if a certain criterion is not satisfied. These devices enormously improve the reliability of the algorithm and good results have been obtained in both cases.

A possible explanation for the success of the symmetric algorithm may lie in the fact that exact line searches are not required in order to achieve quadratic termination. Indeed, as was pointed out by Wolfe [124], it is not even necessary to choose x_{k+1} by the equation (6.3) in order that in the quadratic case, the update should in general give H to be the exact inverse of G after n steps. This is complete contrast with other algorithms of the McCormick-Pearson family and the rank-2 single-parameter family, and is more reminiscent of the secant method. This enables considerable economies to be made when carrying out line searches, and these have contributed materially to the observed success of the algorithm.

Algorithm 5: Powell's Algorithm for Sums of Squares [96]

This method is one in which the objective function $f(x)$, which is given as a sum of squares, that is,

$$f(x) = g(x)^T g(x), \tag{6.39}$$

where

$$g : \mathbb{R}^n \to \mathbb{R}^n,$$

and where it is assumed that $G(x)$, the second derivative of $f(x)$, is not explicitly available. The method is essientially Newton's method, but using a modified secant update to obtain an approximation to $G(x)$. Instead of requiring the approximate matrix B_{k+1} to satisfy (6.10), however, Powell requires that

$$B_{k+1}s_k = d_k\|s_k\|, \tag{6.40}$$

where d_k is a more sophisticated approximation to the directional derivatives along s_k at x_{k+1} than $\frac{y_k}{\|s_k\|}$. Since d_k is computed on the assumption that $f(x)$ is minimized along s_k exact line searches are required. In practice, the algorithm has performed successfully on a large number of problems, although it has been known to converge prematurely to a non-solution [99].

Algoirthm 6: Powell's Hybrid Algorithm [100]

This algorithm is intended for solving the non-linear simultaneous equations $\nabla f(x) = 0$ and it is not applicable to the general non-linear least squares problem. It is thus property outside the scope of the review, but since it uses a least-squares approach we accord it a brief mention. If we define $f(x)$ by $g(x)^T g(x)$ with $g : \mathbb{R}^n \to \mathbb{R}^n$, and apply the steepest descent method to $f(x)$, our direction of search is along the direction $-G(x)^T \nabla f(x)$, where $G(x)$ is the second derivative of $f(x)$. If, on the other hand, we elect to try a Newton step to solve $\nabla f(x) = 0$ our direction of search is along $-G^{-1}(x)\nabla f(x)$. Powell therefore suggests using a linear combination of these two directions as a search direction and since the second derivative matrix G is not assumed to be available, he approximates both $G(x)$ and $G^{-1}(x)$, and generally updates these approximations by the equations (6.27) and (6.28). The final form of the algorithm embodies many checks and safeguards to ensure convergence and stability.

6.4.2 Double-Rank Algorithms

We now consider the double-rank algorithms. All of these are intended to be used only for the function minimization so that no disadvantages accrues from the symmetry of the approximation. All of them, in fact, may be generated by symmetrizing the appropriate rank-1 algorithm.

Algorithm 1: Powell's Symmetric Algorithm [101]

This was the first algorithm to be obtained using a symmetrization technique, which Powell applied as follows. Let B_k be symmetric and obtain $(\bar{B}_1)_{k+1}$ by

using Broyden's formula

$$(\bar{B}_1)_{k+1} = B_k - (B_k s_k - y_k)\frac{s_k^T}{s_k^T s_k}. \tag{6.41}$$

Since $(\bar{B}_1)_{k+1}$ is not symmetric, Powell defines $(B_1)_{k+1}$ by

$$(B_1)_{k+1} = \frac{1}{2}\left((\bar{B}_1)_{k+1} + (\bar{B}_1)_{k+1}^T\right). \tag{6.42}$$

However, $(B_1)_{k+1}$ does not satisfy

$$(B_1)_{k+1} s_k = y_k, \tag{6.43}$$

so that $(B_1)_{k+1}$ is corrected using the equation (6.41) with B_k replaced by $(B_1)_{k+1}$.

The resulting matrix may then by symmetrized by an equation analogous to the equation (6.42) to give $(B_2)_{k+1}$, and the whole process repeated indefinitely. The sequence of matrices $(B_i)_{k+1}$ converges as $i \to \infty$ to the limit B_{k+1}, where

$$B_{k+1} = B_k + \frac{1}{s_k^T s_k}\left[(y_k - B_k s_k)s_k^T + s_k(y_k - B_k s_k)^T\right] - \frac{s_k s_k^T (y_k - B_k s_k)s_k^T}{(s_k^T s_k)^2}, \tag{6.44}$$

and this gives a new updating formula for the approximation to the second derivative matrix G that both satisfies the equation (6.10) and is symmetric. In order to avoid the solution of the equations H_k, the inverse of B_k, is stored and updated by a double-rank formula derived from the equation (6.44).

A feature of this method is the behaviour of the matrix error when the algorithm is applied to the quadratic function defined by the equation (6.5). If the matrix E_k is defined by the equation (6.20), then for this algorithm,

$$E_{k+1} = P_k E_k P_k, \tag{6.45}$$

where

$$P_k = I - \frac{s_k s_k^T}{s_k^T s_k}.$$

Therefore, for the single-rank update in Algorithm 1 in the previous section, the matrix error is post-multiplied by a projection matrix, whereas for the double-rank variation the error is both post-multiplied and pre-multiplied by the same projection matrix. It may be shown that [32] the update is of bounded deterioration and is stable if B_k approximates sufficiently closely to G_k. Algorithms based on using this update with the equation (6.13) do not possess the property of the quadratic termination for any choice of α_k so that there is no theoretical reason for using exact line searches. Indeed, there is evidence that [32] if $f(x)$ is minimized at each step, the update behaves rather badly. If α_k is chosen to be unity for all k, the performance of the update is comparable with that of Broyden in Algorithm 1 in the previous, and the expectation that it would prove superior due to a greater reduction of the norm of E_k at each stage has so far proved to be unfounded. In the algorithm proposed by Powell, as in Algorithm 6 in the previous section, there are many checks and safeguards incorporated to guarantee stability and convergence, but as the algorithm is comparatively recent, there have as yet been few comparisons with other algorithms.

Algorithm 2: The Single-Parameter Rank-2 Family Algorithm [16]

This family includes most of the better-known optimization algorithms, for example, the Davidon-Fletcher-Powell (DFP) algorithm, as special cases. It is that sub-class of Huang family [69], where the correction to H_k is symmetric and rank 2, and where H_{k+1} is constrained to satisfy the equation

$$H_{k+1} y_k = s_k. \tag{6.46}$$

This is, since $H = B^{-1}$, a re-expression of the equation (6.10). Since H_k is symmetric for all k, then the equation for x_{k+1} given by Pearson [90] and Huang

[69], that is, the equation (6.34), reduces to the equation (6.13). Provided that x_{k+1} is always calculated using this equation, and that exact line searches are employed, the successive steps s_j are conjugate for the quadratic functions, and we have the quadratic termination for all algorithms included in the family. It has recently been shown that [20] certain members of the class enjoy the property of bounded deterioration and that if they are employed to minimize the quadratic functions with $\alpha_k = 1$ for all k, the vector errors satisfy the equations similar to the equation (6.30). Moreover, if these algorithms are applied with $\alpha_k = 1$ to more general functions, local convergence proofs may be derived. The case when exact line searches are used has been analysed extensively by Powell ([97], [98]), who proved that the DFP algorithm converges if the objective function $f(x)$ is convex, and converges super-linearly if $f(x)$ is uniformly convex. That these proofs extend to nearly all algorithms of the Huang class follows from a remarkable theorem proved by Huang [69] for the case where $f(x)$ is quadratic, postulated by Huang and Levy [70] for a general $f(x)$ and proved for general $f(x)$ by Dixon [36]. Dixon showed that the nearly all algorithms in the Huang class that satisfy the equation (6.46) will produce the identical sequence of x_k for any arbitrary function, given the identical initial condition and the exact line searches. Therefore, it is only necessary to establish convergence for one algorithm to have demonstrated convergence for the class as a whole.

The updating equation for the rank-2 family is:

$$H_{k+1} = H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \frac{s_k s_k^T}{s_k^T y_k} + \rho_k (H_k y_k - s_k \theta_k)(H_k y_k - s_k \theta_k)^T, \qquad (6.47)$$

where

$$\theta_k = \frac{y_k^T H_k y_k}{s_k^T y_k}$$

and ρ_k is arbitrary. It was shown by Broyden [17] that if H_k is positive-definite, then H_{k+1} is also positive definite for $\rho_k \geq 0$ provided that an exact line search

has been carried out. This result was extended by Shanno, who proved that H_{k+1} is positive definite for $\rho_k > -\delta_k$, where δ_k is some positive number whose value depends upon H_k, y_k and s_k. Since these results require only that $f(x)$ and its gradient are continuous, they imply that algorithms of this family for which $\rho_k \geq 0$ should be stable for all reasonable problems, since all divisors occurring in the equation (6.47) are theoretically positive. This prediction has in general been borne out in practice. The algorithm of Davidon [29] as modified by Fletcher and Powell [50], for which $\rho_k = 0$ for all k, has been used extensively and in general has shown acceptable stability properties. A more recent update ([58], [46], [19], [109]) is obtained when ρ_k is given by

$$\rho_k = \frac{1}{y_k^T H_k y_k}. \tag{6.48}$$

It follows that ρ_k is positive if H_k is positive definite, and this algorithm is also generally stable although like the DFP algorithm, there have been instances where it has failed to converge, probably due to H_k becoming singular. Various explanations have been offered for this departure from the theoretically predicted behaviour, and in particular Bard [5] pointed out that poor scaling could cause H_k to become singular. However, a recent work, by Abbott [1], would appear to indicate that a more probable cause of loss of positive definiteness is failure to perform an exact line search. He showed that if $\rho_k \geq 0$ and H_k is positive definite a necessary and sufficient condition for H_{k+1} to be positive definite is that

$$s_k^T y_k > 0, \tag{6.49}$$

and demonstrated that commonly-used line-search procedures and termination criteria could cause inequality (6.49) to be violated when solving certain problems involving penalty functions. The importance of an exact line-search is underlined by Dixon's theorem. Different members of the class have given in practice widely different results for the same problem starting with the same initial conditions, but a series of very careful experiments by Huang and Levy [70] showed that

these discrepancies disappear when sufficient care is taken in minimizing $f(x)$. We thus see that for members of the family in this algorithm not only are exact line searches necessary for the quadratic termination, but they are also desirable for stability. For this and other reasons, it is possible that algorithms similar to Algorithm 1 above might be increasingly used for unconstrained minimization problems.

6.4.3 Other Applications

We mention here very briefly some more recent algorithms that use quasi-Newton techniques.

Application 1: Brown and Dennis Algorithm [14]

This method is suggested by Brown and Dennis, which is solving the non-linear least squares problem and seems particularly suited for the case where the minimum sum of squares is large. In this method, the components of the second derivative of $f(x) = g(x)^T g(x)$, where $g : \mathbb{R}^n \to \mathbb{R}^n$, are approximated is identical in both cases the latter is preferred since storage is reduced due to the symmetry of the matrices concerned.

Application 2: Dennis Algorithm [33]

This algorithm is used for solving the unconstrained optimization problem. It was first obtained by symmetrization, but instead of symmetrizing Broyden's update given by the equation (6.27) and inverting the resulting formula to give the update for H, Dennis first inverted the equation (6.27) to give the equation (6.28) and symmetrized this. Since the operations of symmetrization and inversion do

not commute Dennis arrived at a new formula where the update is given by

$$H_{k+1} = H_k - \frac{1}{s_k^T H_k y_k}\left[v_k s_k^T H_k + H_k s_k v_k^T - H_k s_k \rho_k s_k^T H_k\right] \tag{6.50}$$

where

$$v_k = H_k y_k - s_k$$

and

$$\rho_k = \frac{v_k^T y_k}{s_k^T H_k y_k}.$$

Dennis found that the performance of this update with $\alpha_k = 1$ was comparable with Powell's symmetric algorithm in Method 1 of the previous section, and if exact line searches were carried out, it still performed quite well whereas Powell's symmetric algorithm failed. As the experimental testing was severely limited, Dennis made no claims for the method other than that it merited further consideration.

Application 3: Constrained Minimization Problem [117]

Further applications of the quasi-Newton principle appear in connection with the constrained minimization problem. In minimizing $f(x)$ subject to the $m(< n)$ equality constraints $c(x) = 0$, the $m + n$ equations to be solved are:

$$J(x)^T z = \nabla f(x) \tag{6.51}$$

subject to

$$c(x) = 0, \tag{6.52}$$

where $J(x)$ is the Jacobian matrix of $c(x)$ and z is the vector (or order m) of Lagrangian multipliers. The Jacobian matrix of the system (6.51) and (6.52) is

$$J_s(x) = \begin{bmatrix} G & J^T(x) \\ J(x) & 0 \end{bmatrix}, \tag{6.53}$$

where G is a linear combination of the second derivative matrices of $f(x)$ and each individual constraint function. there is no reason in principle why the equations (6.51) and (6.52) could not be solved by any quasi-Newton method, but it is hoped that special purpose updates retaining the null partition of $J_s(x)$ might exhibit superior convergence properties. Let

$$\begin{bmatrix} K_k & M_k^T \\ M_k & 0 \end{bmatrix}$$

be the matrix which approximate to $(J_s)_k(x)$. Then, the equation (6.10) may be written as:

$$\begin{bmatrix} K_{k+1} & M_{k+1}^T \\ M_{k+1} & 0 \end{bmatrix} \begin{bmatrix} x_{k+1} - x_k \\ z_{k+1} - z_k \end{bmatrix} = \begin{bmatrix} h_{k+1} - h_k \\ c_{k+1} - c_k \end{bmatrix} \tag{6.54}$$

where

$$h(x) \equiv J(x)^T z - \nabla f(x),$$

and an updating strategy chosen to ensure that this equation is satisfied.

Both the methods discussed below assume that $J(x)$ is explicitly available, though each method uses this information differently:

Method 1: Kwakernaak and Strijbos Method [77]

Set $M_{k+1} = J_{k+1}$. Then, we can obtain, from the equation (6.54), that:

$$K_{k+1}(x_{k+1} - x_k) = h_{k+1} - h_k - J_{k+1}^T(z_{k+1} - z_k). \tag{6.55}$$

Now, K_{k+1}^{-1} is then computed from $(K_k)^{-1}$ using the equation (6.55) and the secant update, and the inverse of the Jacobian of the approximate system is then

obtained using the formula

$$\begin{bmatrix} K & J^T \\ J & 0 \end{bmatrix}^{-1} = \begin{bmatrix} K^{-1} - K^{-1}J^T BJK^{-1} & K^{-1}J^T B \\ BJK^{-1} & -B \end{bmatrix}, \tag{6.56}$$

where

$$B = (JK^{-1}J^T)^{-1}.$$

Method 2: **Broyden and Hart Method** [19]

Use an approximation M_k to J_k when computing the Jacobian of the inverse of the approximate system. Expanding the equation (6.54), gives that:

$$M_{k+1}(x_{k+1} - x_k) = c_{k+1} - c_k \tag{6.57}$$

and

$$K_{k+1}(x_{k+1} - x_k) = h_{k+1} - h_k - M_{k+1}^T(z_{k+1} - z_k) \tag{6.58}$$

and many combinations of update may now be used to compute M_{k+1} and K_{k+1}. A typical, and quite effective one, is to obtain M_{k+1} from the equation (6.57) using the Broyden algorithm in Method 1 in Section 6.4.1 to update and then to compute K_{k+1} using the equation (6.58) and the Powell's symmetric algorithm in Method 1 in Section 6.4.2 to update. Algebraic manipulation then yields a rank-2 updating formula for the approximate inverse system Jacobian.

6.5 Conclusion

Let us conclude the survey of quasi-Newton methods. We have attempted to assess the merits of the basic updates, used either in a "Newton" mode with the steplength $\alpha_k = 1$, that contribute to its observed performance. We have not

attempted to perform a consumer analysis upon the subroutines or procedures in which the updates are used since we are concerned primarily with the update itself. We realize that the reputation of an update may well be due as much to an artful choice of checks, safeguards and program constants with which it is surrounded as to the properties inherent in the update itself, and are only too conscious that a good update may be enhanced, and a poor one disguised, by such devices.

Quasi-Newton methods are widely used and have been extensively treated in the literature, due to the numerous advantages as we described earlier [94]:

(1) At each step, we need the computation of the gradient to update;

(2) There is no inversion, nor solution of a system of linear equations;

(3) The convergence is global;

(4) The convergence rate in a neighborhood of the solution is high, often is a quadratic rate.

On the other hand, yet they are inferior, it needs to store and update an $n \times n$-matrix H_k with significant computer-storage for large n is the greatest disadvantage.

Chapter 7

Choice of Methods in Optimization Problems

7.1 Choice of Methods

In this chapter, we consider the question of the choice of methods, beginning with the case where it is possible to evaluate the derivatives $\frac{\partial f}{\partial x_j}$ without undue difficulty or expense.

The steepest descent method is not recommended for general use because of its poor convergence properties. Newton's method has much better convergence properties; it works particularly well if a close initial estimate of the optimal point can be found. However, it may fail to converge from a poor initial estimate of the optimal point. Also, the evaluation of the elements of the Hessian matrix, and the inversion of this matrix, may pose formidable computational problems.

In the cases where it is difficult or impossible to evaluate the derivatives $\frac{\partial f}{\partial x_j}$,

the DFP method can still be used, with the derivatives approximated by differences. Alternatively, Powell's method may be used, although this method tends to lose its efficiency if the number of variables exceeds fifteen or so, since there is tendency for new directions of search to be chosen less often as the number of variables increases. The possibility of using a direct search method should not be overlooked.

A very useful review paper by Fletcher [44] uses seven different test functions to compare three minimization algorithms that do not require the evaluation of derivatives, namely, the methods of Davies, Swann and Campay [115], Powell [95] and Smith [112].

In 1970, Huang [69] introduced a large three-parameter family of rank 2 algorithms with quadratic termination, which included as special cases the DFP algorithm. The relative merits of the last algorithm is put into perspective by a theorem proved by Huang for quadratic objective functions. He showed that, given the same initial point and the same initial search direction, together with exact linear searches, the same sequence of current points and search directions is generated by every member of his family of algorithms.

Following some numerical experiments by Huang and Levy [70], the corresponding results for non-quadratic objective functions were obtained by Dixon [38]. He showed that, under the above conditions, the sequence of current points and search directions generated by members of Huang's family depends on one parameter only. In particular, the DFP algorithm has the same value of this parameter, and so they produce identical sequences of current points and search directions.

These results suggest that for general purposes the DFP or some closely similar algorithm should be used. Methods involving high-accuracy linear searches can be very time-consuming; hence Powell's method [102], which does not require linear searches, should also be considered.

If $f(x)$ is a sum of squares of functions, as is often the case in data-fitting problems or in the solution of systems of non-linear equations, then special methods are available, for example, the methods of Levenberg [79], Marquardt [81] and Powell [96]. These methods can be expected to converge faster than the general purpose methods considered in this thesis. They are derived from the "Generalized Least Squares Method" which is described at the beginning of Powell's paper [96]. This is an extension of the original "Method of Least Squares" of Legendre and Gauss. Levenberg's and Marquardt's methods are similar, though they were discovered independently. They use the first derivatives of the objective function, while Powell's method requires function evaluations only.

A discussion of some practical points in connection with the implementation of Levenberg's and similar methods is given by Beale [10]. Bard [6] compares the efficiencies of the methods that require the evaluation of derivatives for the maximization of a sum of squares. Finally, Powell gave a comprehensive review of least squares algorithms in Chapter 3: "Problems Related to Unconstrained Optimization" [86].

In my opinion, when solving the optimization problem, if you wish to use the simplest method (just use calculate the first derivatives), then you can choose the steepest descent method. If you wish to have rapid convergence, then you can choose Newton's method (the rate is quadratic). Also, if you wish to have some methods which afford to store the Hessian matrices, then you can choose the other

second derivative methods, for example, Greenstadt's method, numerically stable modified Newton's method, and the like. That is because they afford to store and estimate the inverse of the Hessian matrices. If we solve the optimization problems for the quadratic functions, then you can use the conjugate gradient method since it must be not more than n iterations. But in many practical applications, we will use the Quasi-Newton methods, for example, Secant method, DFP method, and the like. That is because the convergence is global and the rate is often quadratic. Also, it requires the computation on the function values and the gradients of the function.

7.2 Conclusion

At the iterative point $x = x_k$, the function $f(x)$ increases most rapidly in the gradient direction $\nabla f(x_k)$. Gradient methods for minimizing $f(x)$ often use a sequence of linear searches along mutually conjugate directions. The theory of gradient methods is highly developed in the case where the objective function $f(x)$ is quadratic in its arguments, but is less highly developed for more general functions $f(x)$. However, the algorithms derived from 'quadratic' theory may be applied iteratively to non-quadratic objective functions. Some of these algorithms, for example, DFP method, are among the most efficient general purpose optimization techniques available at the present time. Not all gradient methods require analytic expressions for the derivatives. The idea of conjugate directions, or which the most successful methods are based, may also be used when the derivatives cannot be evaluated directly. Powell's method is specifically designed to generate conjugate directions of search without the need to evaluate derivatives; also, quasi-Newton methods with the derivatives replaced by differences have been used successfully. Special methods are available for the minimization of a sum of squares.

Bibliography

[1] J.P. Abbott, *Factors Affecting the Stability of Methods of the Davidon-Fletcher-Powell Type*, in F. Lootsma (ed.), *Numerical Methods for Nonlinear Optimization*, Academic Press, London and New York, 1971.

[2] L. Armijo, *Minimization of Functions Having Lipschitz-Contnuous First Partial Derivatives*, Pacific J. Math. 16, P.1-3, 1966.

[3] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, 1994.

[4] B. Guddat Bank, D.J. Klatte, B. Kummer, *and* K. Tammer, *Non-linear Parametric Optimization*, Akademie-Verlag, Berlin, 1982.

[5] Y. Bard, *On a Numerical Instability of Davidon-like Methods*, Maths Comput. 22, P.665-666, 1968.

[6] Y. Bard, *Comparsion of Gradient Methods for the Solution of Nonlinear Parameter Estimation Problems*, SIAM Journal on Numerical Analysis 7, P.157-186, 1970.

[7] J.G.P. Barnes, *An Algorithm for Solving Nonlinear Equations Based on the Secant Method*, Comput. J. 8, P.66-72, 1965.

[8] Richard Barrett, Michael W. Berry, Tony F. Chan, James Demmel, June Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine, *and* Henk van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods* , SIAM, Philadelphia, 1994.

[9] Jonathan Barzilai *and* Jonathan M. Borwein, *Two-Point Step Size Gradient Methods*, IMA Journal of Numerical Analysis 8, P.141-148, 1988.

[10] E.M.L. Beale, *Numerical Methods*, in Nonlinear Programming (Ed. J. Abadie), North-Holland, Asterdam, Chapter VII, 1967.

[11] E.M.L. Beale, *Introduction to Optimization*, John Wiley and Sons, 1988.

[12] Ernesto G. Birgin, Jose Mario Martinez *and* Marcos Raydan, *Nonmonotone Spectral Projected Gradient Methods on Convex Sets*, SIAM J. Optim., Vol. 10, No. 4, P.1196-1211, 2000.

[13] K. Brown *and* S. Conte, *The Solution of Simultaneous Nonlinear Equations*, Proc. 22 Nat. Conf. ACM. Thomson Book Co., Washington DC, P.111-114, 1967.

[14] K.M. Brown *and* J.E. Dennis, *New Computational Algorithms for minimizing a Sum of Squares of Nonlinear Functions*, Yale University Report 71-6, 1971.

[15] C.G. Broyden, *A Class of Methods for Solving Nonlinear Simultaneous Equations*, Maths. Comput. 19, P.577-593, 1965.

[16] C.G. Broyden, *Quasi-Newton Methods and Their Application to Function Minimization*, Maths. Comput. 21, P.368-381, 1967.

[17] C.G. Broyden, *The Convergence of a Class of Double-Rank Minimization Algorithms: Part 1 and Part 2*, J. Inst. Maths. Applics 6, P.76-90, P.222-231, 1970.

[18] C.G. Broyden, *The Convergence of Single Rank Quasi-Newton Methods*, Maths Comput. 24, P.365-382, 1970.

[19] C.G. Broyden *and* W.E. Hart, *A New Algorithm for Constrained Optimization*, Presented at the 7th Mathematical Programming Symposium, The Hague, The Netherlands, 1970.

[20] C.G. Broyden, J.E. Dennis *and* J.J. More, *On Local and Superlinear Convergence of Quasi-Newton Methods*, J. Inst. Math. Appl. 12, P.223-246, 1976.

[21] A. Cauchy, *Methode Generale Pour La Resolution Des Systems D'equations Simultanees*, Comp. Rend. Acad. Sci. Paris, P.536-538, 1847.

[22] P. Concus, G.H. Golub *and* D.P. O'Leary, *A Generalized Conjugate Gradient Method for the Numerical Solution of Elliptic Differential Equations*, in J.R. Bunch *and* D.J. Rose (ed.), *Sparse Matrix Computations*, Academic Press, New York, P.309-332, 1972.

[23] Y.H. Dai, *New Properties of a Nonlinear Conjugate Method*, Numer. Math. 89, No. 1, P.83-98, 2001.

[24] Y.H. Dai *and* Y. Yuan, *A Nonlinear Conjugate Gradient Method with a Strong Global Convergence Property*, SIAM J. Optim., Vol. 10, No. 1, P.177-182, 1999.

[25] J.W. Daniel, *The Conjugate Gradient Method for Linear and Nonlinear Operator Equations*, SIAM J. Numer. Anal. 4, P.10-26, 1967.

[26] J.W. Daniel, *Approximation Minimization of Functional*, Englewood Cliffs, N.J., Prentice-Hall, 1971.

[27] Yu. M. Danilin, *On a Class of Minimization Algorithms with Over-linear Convergence*, USSR Comput. Maths. Math. Phys. 14, 3, P.59-71, 1974.

[28] W.C. Davidon, *Variance Algorithms for Minimization*, Comput. J. 10, P.406-410, 1968.

[29] W.C. Davies, *Variable Metric Method for Minimization*, AEC Research and Development Systems, Ops Res. 9, P.169-184, 1959.

[30] T.J. Dekker, *Finding a Zero by Means of Successive Linear Interpolation*, in B. Dejon *and* P. Henrici (ed.), *Constructive Aspects of the Fundamental Theorem of Algebra*, Wiley-Interscience, New York, P.37-48, 1969.

[31] N.Y. Deng, *Compuatational Methods for Unconstrained Optimization*, Science Press, Beijing, 1982. (Chinese)

[32] J.E. Dennis, *On the Convergence of Broyden's Method for Nonlinear Systems of Equations*, Maths Comput. 25, P.559-567, 1971.

[33] J.E. Dennis, *On Some Methods Based on Broyden's Secant Approximation to the Hessian* F. Lootsma (ed.), *Numerical Methods for Nonlinear Optimization*, Academic Press, London and New York, 1972.

[34] J.E. Dennis *and* Robert B. Schnabel Jr., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Englewood Cliffs, N.J., Prentice Hall, 1983.

[35] Jr. J. E. Dennis *and* Robert B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, SIAM, Philadelphia, 1996.

[36] L.C.W. Dixon, *Variable Matrix Algorithms: Nececessary and Sufficient Conditions for Identical Behaviour on Nonquadratic Functions*, NOC Technical Report 26, 1971.

[37] L.C.W. Dixon, *Nonlinear Optimisation*, The English Univeities Press Limited, 1972.

[38] L.C.W. Dixon, *Variable Metric Algorithms: Necessary and Sufficient Conditions for Identical Behaviour of Nonquadratic Functions*, J. of Optimization Theory and Applications 10, P.34-40, 1972.

[39] D.K. Faddeev *and* V.N. Faddeeva, *Computational Methods of Linear Algebra*, San Francisco, Freeman, 1963.

[40] A.V. Fiacco, *Introduction to Sensitivity and Stability Analysis in Non-linear Programming*, Academic Press, New York, 1983.

[41] A.V. Fiacco *and* G.P. McCormick, *Computational Algorithm for the Sequential Unconstrained Minimization Technique for Nonlinear Programming*, Mgmt. Sci. 10, P.601-617, 1964.

[42] A.V. Fiacco *and* G.P. McCormick, *Extensions of SUMT for Nonlinear Programming: Equality Constraints and Extrapolation*, Mgmt. Sci 12, P.816-829, 1966.

[43] A.V. Fiacco *and* G.P. McCormick, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, SIAM, Philadelphia, 1990.

[44] R. Flectcher, *Function Minimization Without Evaluating Derivatives - A Review*, The Computer Journal 8, P.33-41, 1965.

[45] R. Fletcher, *A Class of Methods for Nonlinear Programming with Terminal and Convergence Properties*, in J. Abadie (ed.), *Integer and Nonlinear Programming*, North-Holland, Amsterdam, P.157-175, 1970.

[46] R. Fletcher, *A New Approach to Variable Metric Algorithms*, Comput. J. 13, P.317-322, 1970.

[47] R. Fletcher, *Practical Methods of Optimization, Volume 1: Unconstrained Optimization*, John Wiley and Sons, 1980.

[48] R. Fletcher, *Practical Methods of Optimization*, John Wiley and Sons, 2nd edition, 1987.

[49] R. Fletcher, *Low Storage Methods for Unconstrained Optimization*, Lectures in Applied Mathematics, Volume 26, 1990.

[50] R. Fletcher *and* M.J.D. Powell, *A Rapidly Convergent Descent Method for Minimization*, Comput. J. 6, P.163-168, 1963.

[51] R. Flectcher *and* C.M. Reeves, *Function Minimization by Conjugate Gradients*, Comput. J. 6, 2, P.163-168, 1964.

[52] A. Friedlander, J.M. Martinez, B. Molina *and* M. Raydan, *Gradient Method with Retards and Generalizations*, SIAM J. Numer. Anal., Vol. 36, No. 1, P.275-289, 1999.

[53] P.E. Gill *and* W. Murray, *Quasi-Newton Methods for Unconstrained Optimization*, Natn. Phys. Lab. Math. 97, 1971.

[54] P.E. Gill *and* W. Murray, *Newton-Type Methods for Unconstrained and Linearly Constrained Optimization*, Math. Programming 28, P.311-350, 1974.

[55] P.E. Gill *and* W. Murray, *Numerical Methods for Constrained Optimization*, Academic Press, London, New York, 1974.

[56] P.E. Gill *and* W. Murray, *Safeguarded Steplength Algorithms for Optimization Using Descent Methods*, Tech. Rep. NAC 37, National Physical Laboratory Report, Teddington, English, 1974.

[57] P.E. Gill, W. Murray *and* M.H. Wright, *Practical Optmization*, Academic Press, London, 1981.

[58] D. Goldfarb, *A Family of Variable-Metric Methods Derived by Variational Means*, Maths Comput. 24, P.23-26, 1970.

[59] A.A. Goldstein, *Constructive Real Analysis*, Harper and Row, New York, 1967.

[60] G. H. Golub *and* C. F. Van Loan, *Matrix Computations*, The Jason Hopkins University Press, Baltimore, Maryland, 1983.

[61] C.C. Gonzaga, *Two Facts on the Convergence of the Cauchy Algorithm*, Journal of Optimization Theory and Applications, Vol. 107, No.3, P.591-600, 2000.

[62] J.L. Greenstadt, *On the Relative Ineffciencies of Gradient Methods*, Math. Comput. 21, P.360-367, 1967.

[63] J.L. Greenstadt, *Variations of Variable Metric Methods*, Maths Comput. 24, P.1-22, 1970.

[64] R.L. Gue *and* M.E. Thomas, *Mathematical Methods in Operational Research*, Macmillan, New York, 1968.

[65] M.R. Hestenes, *Conjugate Direction Methods in Optimization*, Springer-Verlag, New York, 1980.

[66] M.R. Hestenes *and* E. Stiefel, *Methods of Conjugate Gradients for Solving Linear Systems*, J. Res. Nat. Bur. Stand. USA 49, 6, P.409-436, 1952.

[67] D.M. Himmelblau, *A Uniform Evaluation of Unconstrained Optimization Techniques*, F. Lootsma (ed.), *Numerical Methods for Nonlinear Optimization*, Academic Press, London and New York, 1973.

[68] Y.D. Hu, *Nonlinear Programming*, Higher Education Press, Beijing, 1990. (Chinese)

[69] H.Y. Huang, *Unified Approach to Quadratically Convergent Algorithms for Function Minimization*, J. of Optimization Theory and Applications 5, P.405-423, 1970.

[70] H.Y. Huang *and* A.V. Levy, *Numerical Experiments on Quadratically Convergent Algorithms for Function Minimization*, J. of Optimization Theory and Applications 6, P.269-282, 1970.

[71] V.K. Ivanov, V.V. Vasin, *and* V.P. Tanana, *Theory of Linear Ill-posed Problems and its Applications*, Nauka, Moscow, 1978.

[72] D. Jacobson *and* W. Oksman, *An Algorithm That Minimizers Homogeneous Function of n Variables in n + 2 Iterations and Rapidly Minimizers General Functions*, J. Math. Anal. Appl. 38, 3, P.535-552, 1972.

[73] Karmanov, *Mathematical Programming*, Nauka, Moscow, 1975.

[74] C.T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia, 1995.

[75] C.T. Kelley, *Iterative Methods for Optiminization*, SIAM, Philadelphia, 1999.

[76] J. Kowalik *and* M.R. Osborne, *Methods for Unconstrained Optimization Problems*, American Elsevier Publishing Company, Inc., New York, 1968.

[77] H. Kwakernaak *and* R.C.W. Strijbos, *Extremization of Functions with Equality Constraints*, International Symposium of Mathematical Programming VII, The Hague, 1970.

[78] Lancelot, *A Fortran Package for Large-Scale Nonlinear Optimization (Release A)*, Springer Series in Computational Mathematics, Springer-Verlag, Heidelberg, Berlin, New York, 1992.

[79] K. Levenberg, *A Method for the Solution of Certain Nonlinear Problems in Least Squares*, Quart. Appl. Maths. 2, P.164-168, 1944.

[80] G.I. Marchuk, *Methods of Numerical Mathematics*, Berlin Heidelberg, Springer-Varlag, New York, 1975.

[81] D.W. Marquardt, *An Algorithm for Least-Squares Estimation of Nonlinear Parameters*, J. Soc. Indust. Appl. Math. 11, P.431-441, 1963.

[82] A. Matthews *and* D. Davies, *A Comparison of Modified Newton Methods for Unconstrained Optimization*, ICI Ltd., Central Management Services Research Note, 1969.

[83] A. Matthews *and* D. Davies, *A Comparison of Modified Newton Methods for Unconstrained Optimization*, Comput. J. 14, P.293-294, 1971.

[84] J.J. More, *The Levenberg-Marquardt Algorithm: Implementation and Theory*, in Numerical Analysis, G.A. Watson (ed.), *Lecture Notes in Mathematics* 630, Springer-Verlag, Berlin, P.105-116, 1977.

[85] J.J. More *and* S.J. Wright, *Optimization Software Guide*, SIAM, Philadephia, 1983.

[86] W. Murray, *Numercial Methods for Unconstrained Optimization*, Academic Press, London, 1972.

[87] W. Murray *and* M.L. Overton, *Steplength Algorithms for Minimizing a Class of Nondifferentiable Functions*, Computing 23, P.309-331, 1979.

[88] B.A. Murtagh *and* R.H.W. Sargent, *A Constrained Minimization Method with Quadratic Convergence*, in R. Fletcher (ed.), *Optimization*, Academic Press, London and New York, 1970.

[89] J.M. Ortega *and* W.C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[90] J.D. Pearson, *Variable Metric Methods of Minimization*, Comput. J. 12, P.171-178, 1969.

[91] E. Polak, *Computational Methods in Optimization; A Unified Approach*, Academic Press, New York, 1971.

[92] Boris T. Polyak, *Some Methods of Speeding Up the Convergence of Iteration Methods*, USSR Comput. Maths. Math. Phys. 4, 5, P.1-17, 1964.

[93] Boris T. Polyak, *The Conjugate Gradient Method in Extremal Problems*, USSR Comput. Maths. Math. Phys. 9,4, P.94-112, 1969.

[94] Boris T. Polyak, *Introduction to Optimization*, Optimization Software Inc. Publications Division, New York, 1987.

[95] M.J.D. Powell, *An Efficient Method of Finding the Minimum of a Function of Several Variables Without Calculating Derivatives*, The Computer Journal 7, P.155-162, 1964.

[96] M.J.D. Powell, *A Method for Minimizing a Sum of Squares of Nonlinear Functions Without Calculating Derivatives*, The Computer Journal 7, P.303-307, 1965.

[97] M.J.D. Powell, *A Method for Nonlinear Constraints in Minimization Problems*, in R. Fletcher (ed.), *Optimization*, Academic Press, London and New York, P.283-293, 1969.

[98] M.J.D. Powell, *A Theorem on Rank One Modification to a Matrix and Its Inverse*, Comput. J. 12, P.288-290, 1969.

[99] M.J.D. Powell, *A FORTRAN Subroutine for Solving Systems of Nonlinear Algebraic Equations*, in P. Rabinowitz (ed.), *Numerical Methods for Nonlinear Algebraic Equations*, Gordon and Breach, P.115-161, 1970.

[100] M.J.D. Powell, *A Hybrid Method for Nonlinear Equations*, in P. Rabinowitz (ed.), *Numerical Methods for Nonlinear Algebraic Equations*, Gordon and Breach, P.87-114, 1970.

[101] M.J.D. Powell, *A New Algorithm for Unconstrained Optimization*, Report TP 393, AERE R.6469, 1970.

[102] M.J.D. Powell, *Quadratic Termination Properties of Minimization Algorithms. I Statement and Discussion of Results*, J. Inst. Maths. Applics. 10, P.333-342, 1972.

[103] B.M. Pshenichnyj *and* Yu.M. Danilin, *Numerical Methods in Extremal Problems*, Mir., Moscow, 1978.

[104] M. Raydan, *On the Barzilai and Borwein Choice of Steplength for the Gradient Method*, IMA Journal of Numerical Analysis 13, P.321-326, 1993.

[105] M. Roma, *Large Scale Unconstrained Optimization*, in C. Floudas *and* P. Pardalos (ed.) *Encyclopedia of Optimization*, Technical Report 10-99, Kluwer Academic Publishers, 1999.

[106] W. Rudin, *Principles of Mathematical Analysis*, McGraw-Hill, New York, 3rd edition, 1976.

[107] V. Samanski, *On the Modification of the Newton Method*, Ukrain, Math. Z. 19, P.218-227, Russian, 1967.

[108] R.B. Schnabel *and* E. Eskow, *A New Modified Cholesky Factorization*, SIAM J. Sci. Statist. Comput., 11, P.1136-1158, 1990.

[109] D.F. Shanno, *Conditioning of Quasi-Newton Methods for Function Minimization*, Maths Comput. 24, P.647-656, 1970.

[110] S.L. Shi, *Nonlinear Optimization Methods*, Higher Education Press, Beijing, 1992. (Chinese)

[111] S.L. Shi *and* F.J. Jau, *Numerical Optimization Methods*, Shenghai Scientific Technology Press, Shenghai, 1983. (Chinese)

[112] C.S. Smith, *The Automatic Computation of Maximum Likelihood Estimates*, N.C.B. Scientific Dept., Repaort No. SC846/MR/40, 1962.

[113] J.A. Snyman *and* A.M. Hay, *The Spherical Quadratic Steepest Descent (SQSD) Method for Unconstrained Minimization with No Explicit Line Searches*, Computers and Mathematics with Applications 42, P.169-178, 2001.

[114] G.W. Stewart, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

[115] W.H. Swann, *Report on the Development of a New Direct Search Method of Optimization*, I.C.I. Ltd., Central Instrument Laboratory Research Note 64/3, 1964.

[116] A.N. Tikhonov *and* V.Y. Arsenin, *Solutions of Ill-posed Problems*, W.H. Winston, Washington, D.C., 1977.

[117] X.J. Tong *and* S. Zhou, *A Trust-Region Augion Augmented Lagrangian Algorithm for Equality and Bounded Constraints*, Mathematica Numerica Sinica, Vol. 24, No. 1, Changsha, 2002. (Chinese)

[118] Vasil'ev, *Lectures on the Methods for Solving Extremal Problems*, Izd. MGU, Moscow, 1974.

[119] M.N. Vrahatis, G.S. Androulakis, J.N. Lambrinos *and* G.D. Magoulas, *A Class of Gradient Unconstrained Minimization Algorithms with Adaptive Stepsize*, Journal of Computational and Applied Mathematics 114, P.367-386, 2000.

[120] G.R. Walsh, *Methods of Optimization*, John Wiley and Sons Ltd., Revised Edition, 1979.

[121] J.H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, London, 1965.

[122] S.J. Wright *and* J.N. Holt, *An Inexact Levenbert-Marquardt Method for Large Sparse Nonlinear Least Squares*, J. Austral. Math. Soc. Ser. B 26, P.387-403, 1985.

[123] P. Wolfe, *The Secant Method for Simultaneous Nonlinear Equations*, Ass. Comput. Math. Commun. 2, 12, P.12-13, 1959.

[124] P. Wolfe, *Another Variable Metric Method*, Working Paper, 1967.

[125] P. Wolfe, *Convergence Conditions for Ascent Methods*, SIAM Rev. 11, P.226-235, 1969.

[126] P. Wolfe, *Convergence Conditions for Ascent Methods II: Some Corrections*, SIAM Rev. 13, P.185-188, 1971.

[127] Y. Yuan, *Numerical Methods for Nonlinear Programming*, Shenghai Scientific Technology Press, Shenghai, 1993. (Chinese)

[128] Y. Yuan *and* W. Sun, *Optimization Theory and Methods*, Science Press, Beijing, 1999. (Chinese)